Quick 5 user manual

A GPU version of CMLAir

Tholfaqar (Dolf) Mardan and David B. Bogy

Computer Mechanics Laboratory

Mechanical Engineering

University of California at Berkeley

June 2013

Table of contents

List of symbols and acronyms

Vocabulary	Meaning
GPU	Graphical Processing Unit
DLL	Dynamic Link Library
NVIDIA	GPU manufacturing company located in San Jose
Cuda Fortran	Version of Fortran able to communicate with GPU
Kernel	Special subroutine in Cuda Fortran to link between Fortran 90 and GPU hardware
Cuda cores	Processors inside the GPU card
GDDR5 memory	Graphics Double Data Rate memory version 5
CC	A number refers to GPU Compute Compatibility

Introduction to GPU Quick 5.0

Quick code version 5.0 is an upgrade from the Quick 4 code. It is more intelligent yet faster than Quick 4. It has its unique internal structure utilizing the computing power of a Graphical Processing Unit (GPU) efficiently for scientific heavily mathematical calculations. The main concept behind the dramatic speed improvement (3x for some complex slider designs) is the division of the tasks into smaller portions and computing them in a synchronized and parallel manner rather than computing serially. Parallel computing has proven its superiority and speed efficiency over sequential computing.

Quick 5.0 has the same input files and same output files as Quick 4. Since it runs on a different hardware structure than the conventional hardware such as computer CPU and Ram it needs a special DLL (dynamic link library) file to run successfully. This DLL is already included in the latest release of CMLAir8.1, the new Quick code designed to run on a compatible NVIDIA GPU. Quick 5.0 can run on a NVIDIA GeForce 680 GPU card or any NVIDIA GPU card with CC (Compute Compatibility) of 3.0 or higher. We will talk about compatible GPU's in detail in a later section of this manual.

The new CMLAir8.1 includes both the Quick 5.0 static solver and the required DLL. There is no need to download them separately, however, when you install CMLAir, the default Quick solver is Quick 4. If you have the proper NVIDIA hardware and software installed in your system and want to use Quick 5.0 instead, you need to go to the solver section in CMLAir and manually select the GPU Quick 5.0 solver.

Features added

The following features and improvements have been added to the Quick code to create the newest and fastest Quick code version 5.0:

- The Quick code version 5.0 has 16,060 lines of code, which includes 4,400 new lines added to the Quick code 4.32 (11,660 lines).
- One file has been added, kernels.f90 which has 2,320 lines of code and 26 kernel subroutines.
- 8 new modules have been added to enhance portability of data handled by the code and optimum memory utilization.
- 2 files have been migrated from Fortran 77 to Fortran 90 to support Cuda Fortran.
- About 67% speed gain has been achieved, as shown in a later section of this manual.

Why GPU and Cuda Fortran?

- GPU supports parallel processing.
- It has more processors than the latest CPU.
- GPU has a more reliable, robust, and easier to use and control multi-processing module than CPU.
- It works separately from the CPU since it has its own hardware.
- It supports MPI implementation of tasks (multi GPU's).
- It is available and more affordable than CPU.
- Cuda C (by NVIDIA) and Cuda Fortran (by Portland group, 2007) are the two languages provided to communicate with NVIDIA GPU's

Why Quick 5.0?

80% of the new Quick code version 5.0 runs on GPU architecture. Furthermore, depending on the shape and initial conditions of the air bearing slider, you can get different speed improvement characteristics compared to the Quick code version 4. This can vary with slider design, as shown in the examples and figures in a later section, from 20% for a traditional and simple or moderately complex air bearing slider designs to as much as 67% for more complex designs.

Computer system requirements:

a. Hardware requirements

During the testing and implementation on the new GPU Quick 5.0, we have used a Dell desktop computer model: XPS 8500. It has powerful core i7-3770 processors, 3.4 GHz (two processors on a chip), 16 GB of DDR 3 Ram and a PCI express slot version 3.0 where you attach the GPU to the motherboard to get maximum speed transferring data in and out of the GPU. A decent power supply on board is required since the GPU, during calculations, consumes a relatively large amount of DC power. Each GPU has its rated maximum power that needs to be noted and accommodated. Inside the Dell XPS 8500 we installed the following NVIDIA graphic card model: GeForce 680, which is a medium level GPU with powerful features and compelling price and is used by many video game enthusiasts. A picture of the GPU is shown in Fig. 1. It has the interesting technical specifications given in Table 1.

Figure 1. NVIDIA GeForce 680 GPU card

Cuda Cores	1536
Base Clock (MHz)	1006
Boost Clock (MHz)	1058
Memory Size (GB)	2
Memory interface width	256-bit GDDR5
Memory bandwidth (GB/sec)	192.2
Interface Bus support	PCI Express 3.0
Power dissipation (Watts)	25 idle, 195 maximum
Minimum power supply size (Watts)	550
Dimensions (Inch)	10 length x 4.38 height x 1.5 width
Price ($US)	450

Table 1. NVIDIA GeForce 680 specifications chart.

As seen in Table 1 above, the GeForce GPU card has quite interesting characteristics that can be utilized for CML math calculations. It has 1536 cores running at 1 GHz speed. In addition it has 2 GB of fast GDDR5 memory with 256-bit attached and divided among the 1536 processors. This unique memory reduces the need for transferring data back and forth between the GPU and CPU and ultimately reduces the heap and obviously speeds up problem solving. Having a PCI express version 3.0 bus takes data transfer to a new level compared to older GPUs. That explains the ultra-fast transfer rate of 192.2 GB/second between the GPU and CPU and its peripherals. On the other hand, such a GPU needs substantial power to perform its operations when running simulations. The reasons why we chose Dell XPS 8500 are: first, it has the minimum power supply required by the GeForce 680, secondly, it has a PCI express 3.0 bus interface on its motherboard, and ultimately, its case and the inside space is more than enough to accommodate comfortably the GeForce 680. It is recommended not to use the DVD/CD for playing or burning while using GPU Quick 5.0 in order to protect the power supply from overheating. No further action is needed in regards to the heat generated by the card when working at its maximum capacity since the GPU has a large centralized fan on top of the main chip specially designed for this task, as illustrated in Figure 1. Price wise, the GeForce 680 is in the medium level between the high end, super expensive Tesla K20 cards (priced at $3,600 US) and low end older GeForce models, such as the GeForce 460 (priced at $120) with fewer and slower processors and smaller memory.

GPU Quick 5.0 has been tested with all of the above previously mentioned GPU architectures. This Includes: GeForce 460, GeForce 680 and lastly Tesla K20. Due to the fact that GPU Quick 5.0 was demonstrated previously on GeForce 460 and the results have been shown in the 25th CML sponsors' meeting, we show only new results of performance tests on only two GPUs: GeForce 680 and Tesla K20 in this manual. The Tesla K20 GPU card is shown in Figure 2. Its corresponding technical data specifications are mentioned in Table 2.

Figure 2. NVIDIA GeForce 680 GPU card

Cuda Cores	2496
Base Clock (MHz)	706
Boost Clock (MHz)	N/A
Memory Size (GB)	5
Memory interface width	320-bit GDDR5
Memory bandwidth (GB/sec)	208
Interface Bus support	PCI Express 2.0
Power dissipation (Watts)	25 idle, 225 maximum
Minimum power supply size (Watts)	650
Dimensions (Inch)	10.5 length x 4.38 height x 1.5 width
Price ($US)	3,600

Table 2. NVIDIA Tesla K20 specifications chart.

Due to the high price of Tesla K20 card plus the special desktop tower needed to run it, we decided not to buy this hardware. Instead, we asked a few engineers from NVIDIA and Exxact (marketing partner for NVIDIA Tesla) Corporations to let us remote log on to their super computer equipped with Tesla K20. With plenty of appreciated support from those engineers, we were able to test GPU Quick 5.0 on their Tesla K20 powered super computer in San Jose.

Their server has the following specifications: 2 Intel Xeon CPUs at speed of 2.2 GHz, 64 GB of Ram and four of Tesla K20 GPUs. Here is a link to their server's information page: http://exxactcorp.com/index.php/solution/solu_detail/90 . We were using only one out of the four GPUs. The GPU Quick 5.0 is currently designed to utilize only one GPU at a time.

b. Software requirements

After obtaining the right hardware, a medium form factor desktop computer such as the Dell XPS 8500 or equivalent computer, a GPU such as the GeForce 680 or above, you can directly and easily install the GPU and replace the already installed generic graphic card. Install the hardware into the desktop computer with the proper precautions such as discharging any electrostatic charges off your hands and trying not to apply too much pressure while installing the GPU into the proper slot, which could lead to damage to the motherboard and off course a malfunctioning GPU.

The GPU card comes with an installation DVD for installing the card drivers, however, it is recommended to go to the www.nvidia.com website and download the latest GPU driver software. This way you are sure of getting the best compatibility and bug free driver software. As of the time of writing this manual, the latest GeForce driver is version 314.20, which is the driver we used when testing the new GPU Quick 5.0. The installation of the latest GPU driver is a straight forward process, just follow the installation instructions and it should take no longer than a few minutes.

Regarding the installation of the GPU Quick 5.0 solver executable, there are two ways to do it: directly by installing the latest CMLAir8.1 which includes GPU Quick 5.0, or via downloading the solver from the CML website along with one required run time Cuda Fortran DLL file. Depending on which GPU you are planning to use, there will be two options of GPU Quick 5.0 solvers to choose from: GeForce or Tesla GPU Quick 5.0 solvers. The DLL file should be the same for both solvers to avoid confusion and reduce unnecessary files to be downloaded.

If you would like to try a different GPU than GeForce 680 or Tesla K20, please let Dolf know. He will have the correct compiled GPU Quick 5.0 that fits the CC requirements for your GPU.

Test results on example sliders

In this section we discuss the example sliders used and the results and corresponding speed improvements obtained by using the GPU Quick 5.0 solver.

Testing the new GPU Quick code version 5.0 was a substantial task. So we tried to use multiple sliders and multiple GPUs, both simple and complex slider designs to ensure variety and accuracy of results. All of the sliders have been tested with medium and high complexity GPUs. In this suction we show 8 sliders as examples for the test. One of them is the Quick 4 example slider that comes with our CMLAir8.1. The rest of the sliders are either sliders recently studied by CML students or from CMLAir users in the hard disk industry. The reason we used the simple CMLAir example file, which does not need GPU Quick 5.0 to solve it, is to show the direct proportion between complexity of design and speed improvement gained when using the new GPU Quick 5.0. All sliders under discussion have been tested on a GeForce 680, on a Tesla K20 GPU as well as using Quick 4, as mentioned in an earlier section. For each example slider, we show the rail design in addition to the corresponding speed performance. Furthermore, since the GeForce 680 and Tesla K20 GPU cards were installed in two different desktop computers with different CPU architecture, we will show two speed performance tables for each example. The first table will be for the

Tesla K20 GPU solver versus the Quick 4 solver on one computer, while the second table will show comparison between GeForce 680 GPU and the Quick 4 solver on a different computer.

Example Sliders:

Slider 1

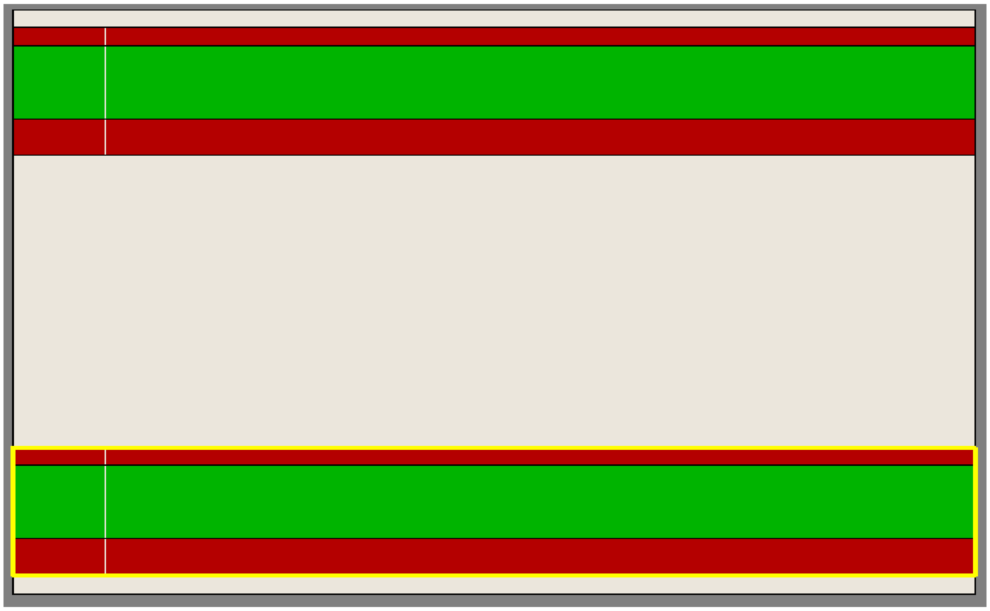


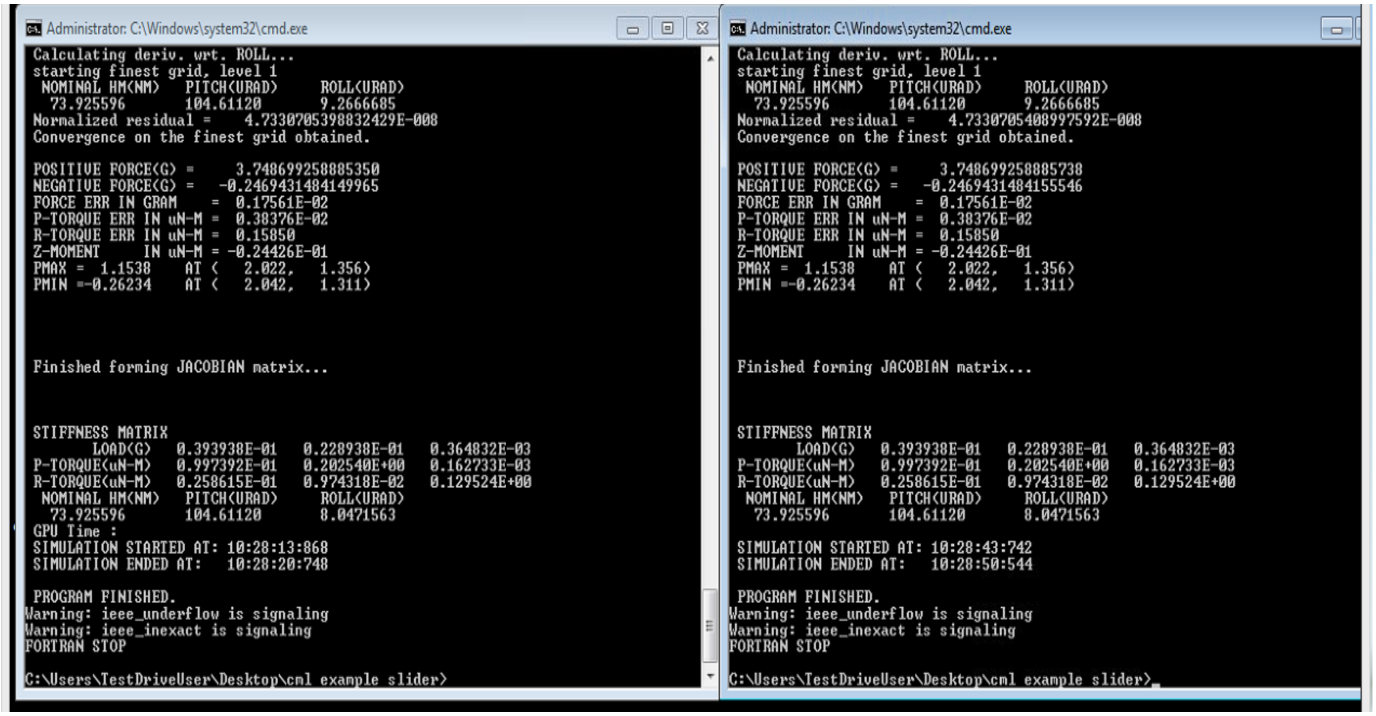
Figure 3a. CMLAir Quick4 example slider rails

Quick Solver type	Tesla K20 GPU	Quick 4	Speed increase
Execution time	7 seconds	7 seconds	0 %

Table 3 a. Tesla K20 GPU solver vs Quick 4 solver

Quick Solver type	GeForce 680 GPU	Quick 4	Speed increase
Execution time	3 seconds	3 seconds	0 %

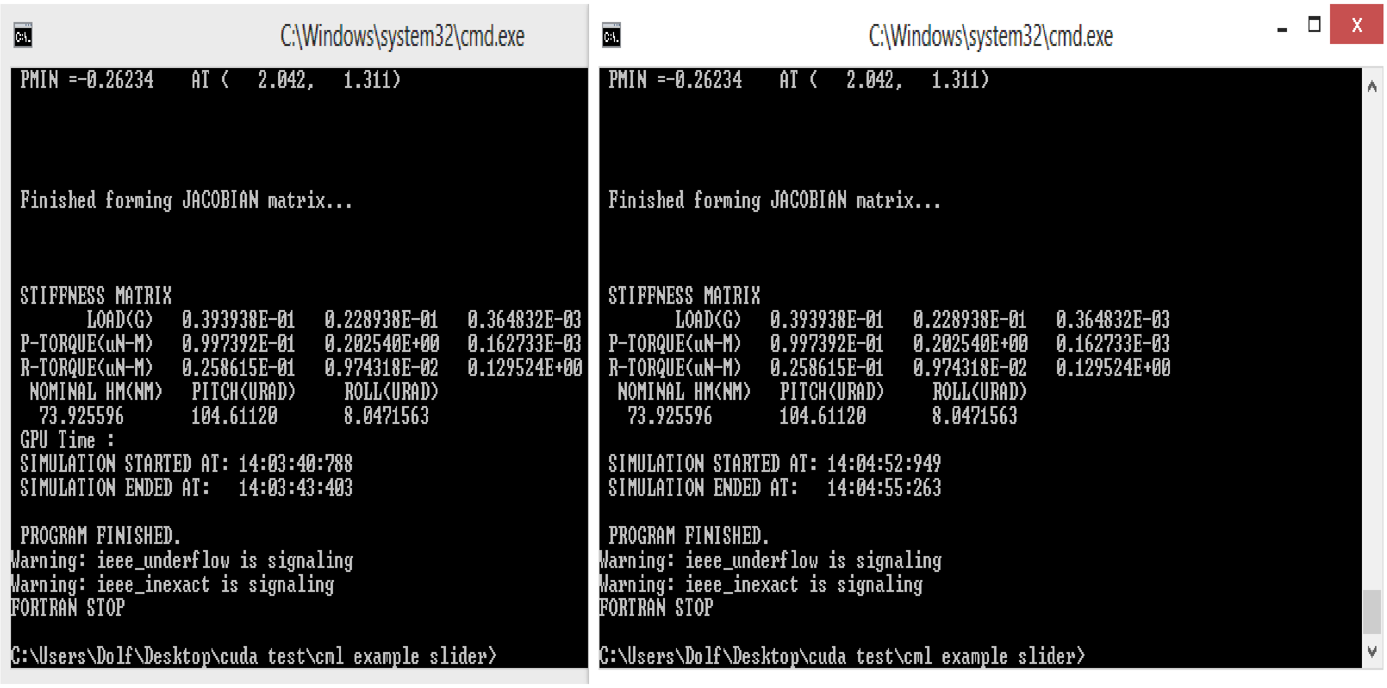
Table 3 b. GeForce GPU solver vs Quick 4 solver



```
Calculating deriv. wrt. ROLL...
starting finest grid, level 1
 NOMINAL HM(NM)   PITCH(URAD)     ROLL(URAD)
  73.925596       104.61120       9.2666685
Normalized residual =    4.7330705398832429E-008
Convergence on the finest grid obtained.

POSITIVE FORCE(G) =     3.748699258885350
NEGATIVE FORCE(G) =   -0.2469431484149965
FORCE ERR IN GRAM    =  0.17561E-02
P-TORQUE ERR IN uN-M =  0.38376E-02
R-TORQUE ERR IN uN-M =  0.15850
Z-MOMENT     IN uN-M = -0.24426E-01
PMAX =  1.1538    AT (   2.022,   1.356)
PMIN =-0.26234    AT (   2.042,   1.311)



Finished forming JACOBIAN matrix...



STIFFNESS MATRIX
       LOAD(G)   0.393938E-01   0.228938E-01   0.364832E-03
P-TORQUE(uN-M)   0.997392E-01   0.202540E+00   0.162733E-03
R-TORQUE(uN-M)   0.258615E-01   0.974318E-02   0.129524E+00
 NOMINAL HM(NM)   PITCH(URAD)     ROLL(URAD)
  73.925596       104.61120       8.0471563
GPU Time :
SIMULATION STARTED AT: 10:28:13:868
SIMULATION ENDED AT:   10:28:20:748

PROGRAM FINISHED.
Warning: ieee_underflow is signaling
Warning: ieee_inexact is signaling
FORTRAN STOP

C:\Users\TestDriveUser\Desktop\cml example slider>
```

```
Calculating deriv. wrt. ROLL...
starting finest grid, level 1
 NOMINAL HM(NM)   PITCH(URAD)     ROLL(URAD)
  73.925596       104.61120       9.2666685
Normalized residual =    4.7330705408997592E-008
Convergence on the finest grid obtained.

POSITIVE FORCE(G) =     3.748699258885738
NEGATIVE FORCE(G) =   -0.2469431484155546
FORCE ERR IN GRAM    =  0.17561E-02
P-TORQUE ERR IN uN-M =  0.38376E-02
R-TORQUE ERR IN uN-M =  0.15850
Z-MOMENT     IN uN-M = -0.24426E-01
PMAX =  1.1538    AT (   2.022,   1.356)
PMIN =-0.26234    AT (   2.042,   1.311)



Finished forming JACOBIAN matrix...



STIFFNESS MATRIX
       LOAD(G)   0.393938E-01   0.228938E-01   0.364832E-03
P-TORQUE(uN-M)   0.997392E-01   0.202540E+00   0.162733E-03
R-TORQUE(uN-M)   0.258615E-01   0.974318E-02   0.129524E+00
 NOMINAL HM(NM)   PITCH(URAD)     ROLL(URAD)
  73.925596       104.61120       8.0471563

SIMULATION STARTED AT: 10:28:43:742
SIMULATION ENDED AT:   10:28:50:544

PROGRAM FINISHED.
Warning: ieee_underflow is signaling
Warning: ieee_inexact is signaling
FORTRAN STOP

C:\Users\TestDriveUser\Desktop\cml example slider>_
```

Figure 3b. Tesla K20 GPU output vs Quick 4 output.



```
PMIN =-0.26234    AT (   2.042,   1.311)



Finished forming JACOBIAN matrix...



STIFFNESS MATRIX
       LOAD(G)   0.393938E-01   0.228938E-01   0.364832E-03
P-TORQUE(uN-M)   0.997392E-01   0.202540E+00   0.162733E-03
R-TORQUE(uN-M)   0.258615E-01   0.974318E-02   0.129524E+00
 NOMINAL HM(NM)   PITCH(URAD)     ROLL(URAD)
  73.925596       104.61120       8.0471563
GPU Time :
SIMULATION STARTED AT: 14:03:40:788
SIMULATION ENDED AT:   14:03:43:403

PROGRAM FINISHED.
Warning: ieee_underflow is signaling
Warning: ieee_inexact is signaling
FORTRAN STOP

C:\Users\Dolf\Desktop\cuda test\cml example slider>
```

```
PMIN =-0.26234    AT (   2.042,   1.311)



Finished forming JACOBIAN matrix...



STIFFNESS MATRIX
       LOAD(G)   0.393938E-01   0.228938E-01   0.364832E-03
P-TORQUE(uN-M)   0.997392E-01   0.202540E+00   0.162733E-03
R-TORQUE(uN-M)   0.258615E-01   0.974318E-02   0.129524E+00
 NOMINAL HM(NM)   PITCH(URAD)     ROLL(URAD)
  73.925596       104.61120       8.0471563

SIMULATION STARTED AT: 14:04:52:949
SIMULATION ENDED AT:   14:04:55:263

PROGRAM FINISHED.
Warning: ieee_underflow is signaling
Warning: ieee_inexact is signaling
FORTRAN STOP

C:\Users\Dolf\Desktop\cuda test\cml example slider>
```

Figure 3c. GeForce 680 GPU solver output vs Quick 4 output.

Slider 2

(slider design not shown for this case)

Figure 4. Slider 2 rails

Quick Solver type	Tesla K20 GPU	Quick 4	Speed increase
Execution time	8 minutes	15 minutes	46 %

Table 4 a. Tesla K20 GPU solver vs Quick 4 solver

Quick Solver type	GeForce 680 GPU	Quick 4	Speed increase
Execution time	6 minutes	9 minutes	33 %

Table 4 b. GeForce GPU solver vs Quick 4 solver

Slider 3

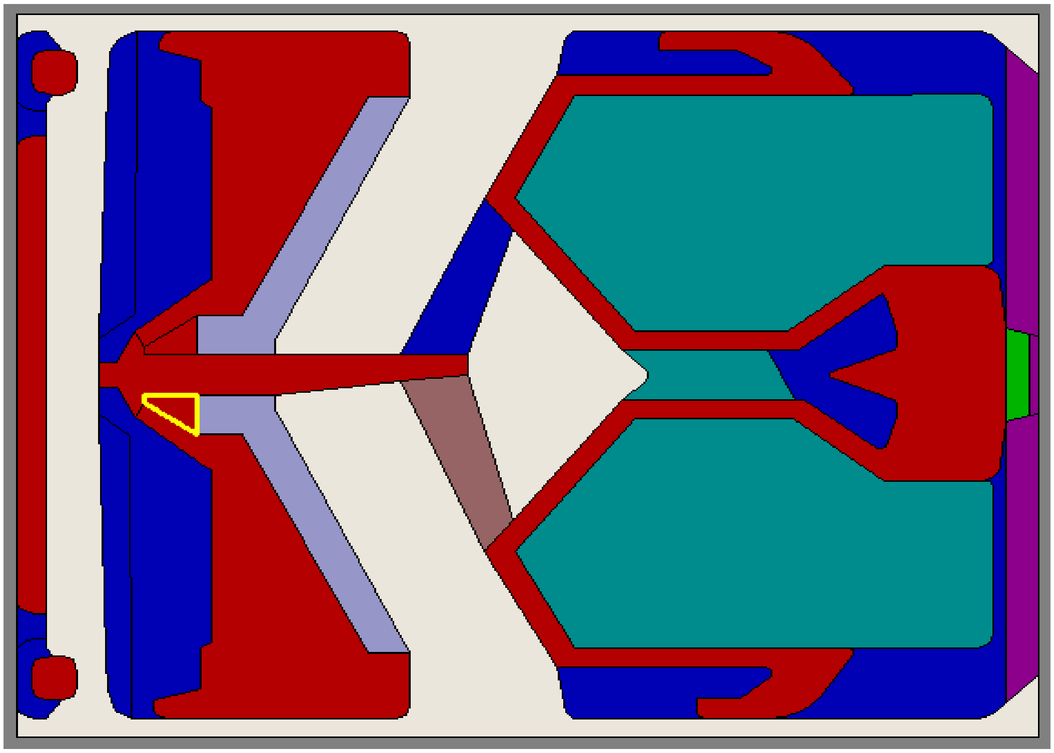


Figure 5. Slider 3 rails

Quick Solver type	Tesla K20 GPU	Quick 4	Speed increase
Execution time	1 minute	3 minutes	66.7 %

Table 5 a. Tesla K20 GPU solver vs Quick 4 solver

Quick Solver type	GeForce 680 GPU	Quick 4	Speed increase
Execution time	1 minute	1 min 40 sec	40 %

Table 5 b. GeForce GPU solver vs Quick 4 solver

Slider 4

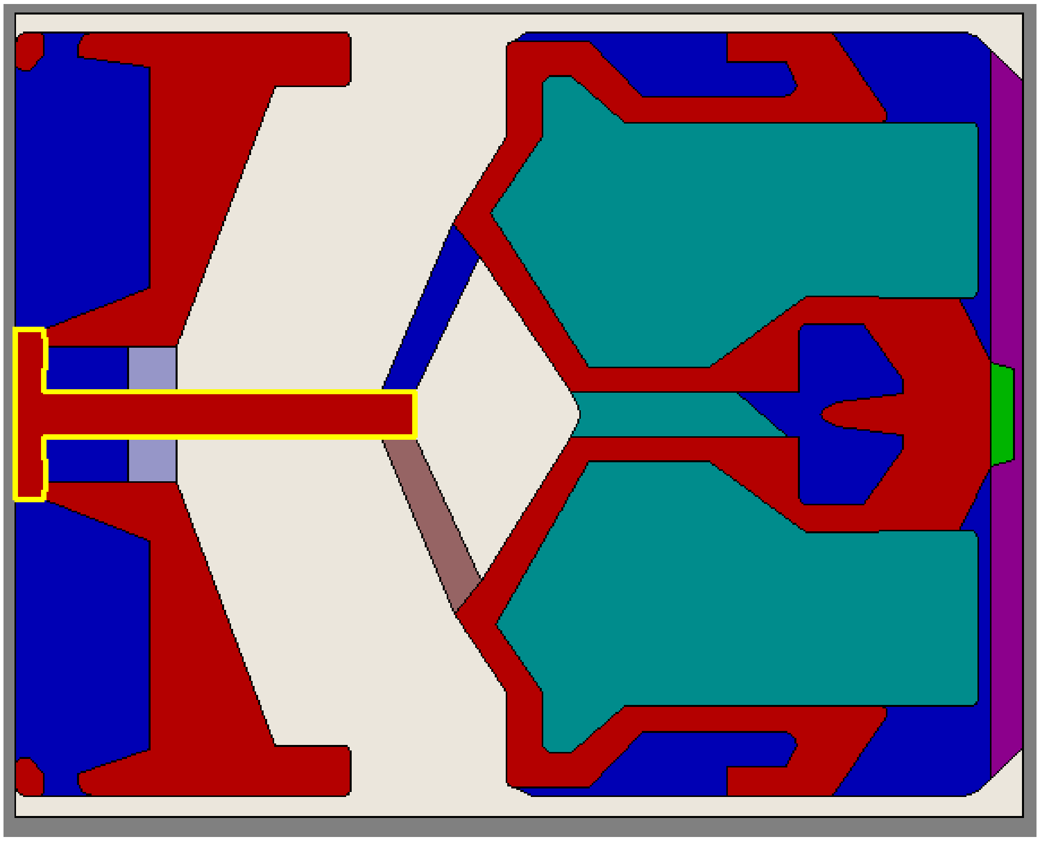


Figure 6. Slider 4 rails

Quick Solver type	Tesla K20 GPU	Quick 4	Speed increase
Execution time	2 minutes	3 minutes	33.3 %

Table 6 a. Tesla K20 GPU solver vs Quick 4 solver

Quick Solver type	GeForce 680 GPU	Quick 4	Speed increase
Execution time	2 minute	3 minutes	33.3 %

Table 6 b. GeForce GPU solver vs Quick 4 solver

Slider 5

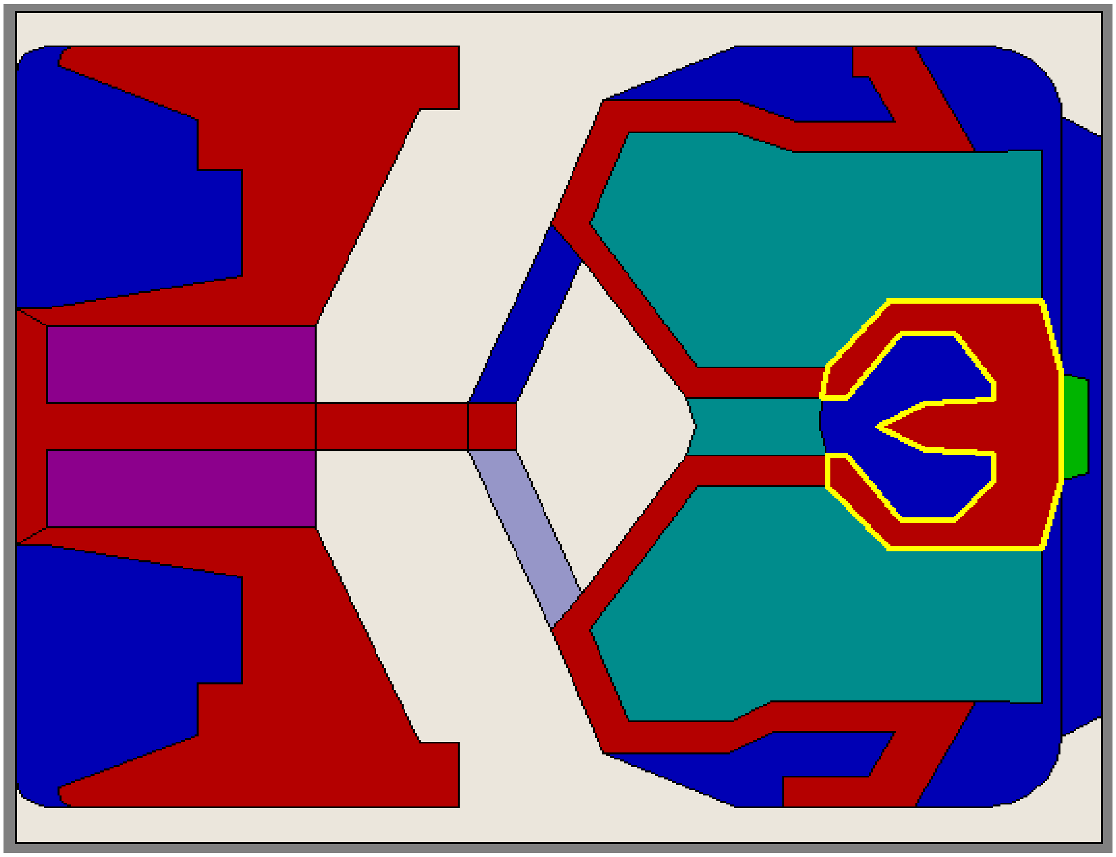


Figure 7. Slider 5 rails

Quick Solver type	Tesla K20 GPU	Quick 4	Speed increase
Execution time	2 minutes	5 minutes	60 %

Table 7 a. Tesla K20 GPU solver vs Quick 4 solver

Quick Solver type	GeForce 680 GPU	Quick 4	Speed increase
Execution time	1 minute	2 minutes	50 %

Table 7 b. GeForce GPU solver vs Quick 4 solver

Slider 6

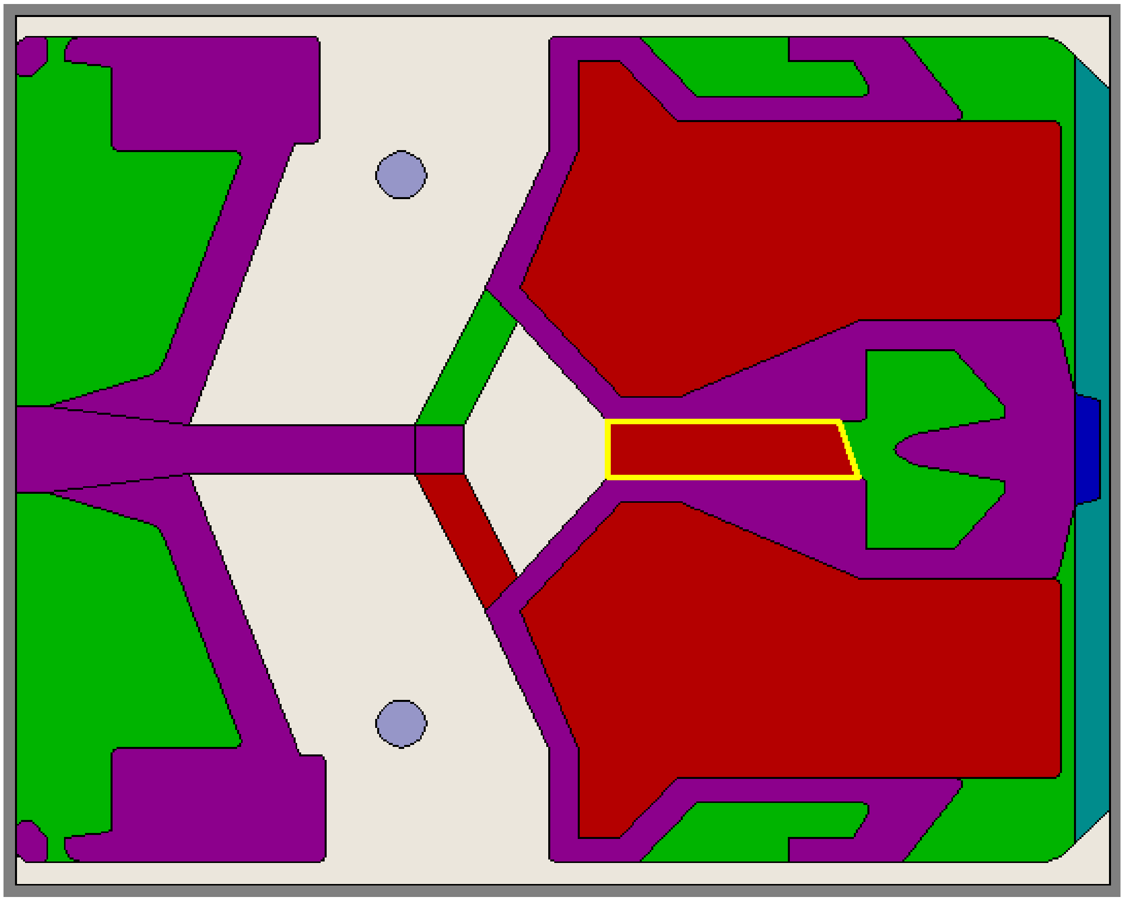


Figure 8. Slider 6 rails

Quick Solver type	Tesla K20 GPU	Quick 4	Speed increase
Execution time	18 minutes	37 minutes	51 %

Table 8 a. Tesla K20 GPU solver vs Quick 4 solver

Quick Solver type	GeForce 680 GPU	Quick 4	Speed increase
Execution time	16 minutes	25 minutes	36 %

Slider 7

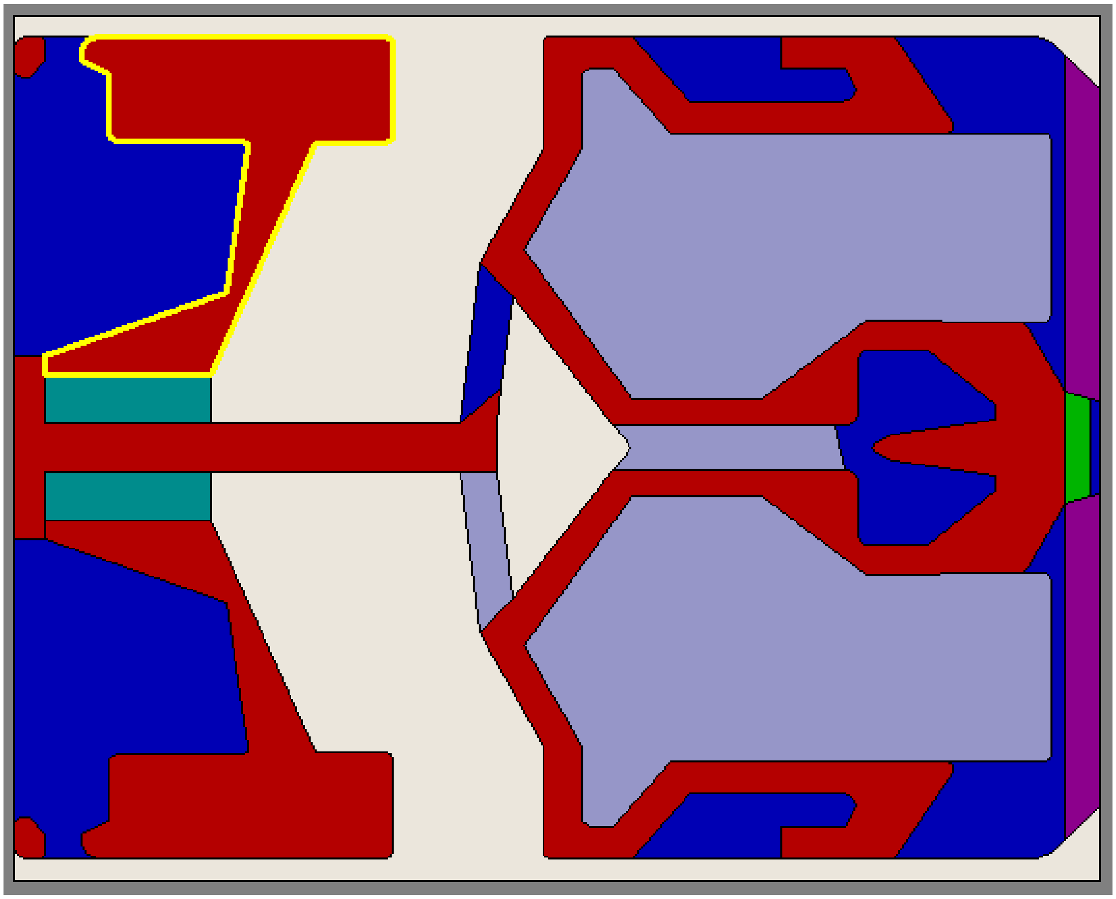


Figure 9. Slider 7 rails

Quick Solver type	Tesla K20 GPU	Quick 4	Speed increase
Execution time	3 minutes	5 minutes	40 %

Table 9 a. Tesla K20 GPU solver vs Quick 4 solver

Quick Solver type	GeForce 680 GPU	Quick 4	Speed increase
Execution time	3 minutes	4 minutes	25 %

Table 9 b. GeForce GPU solver vs Quick 4 solver

Slider 8

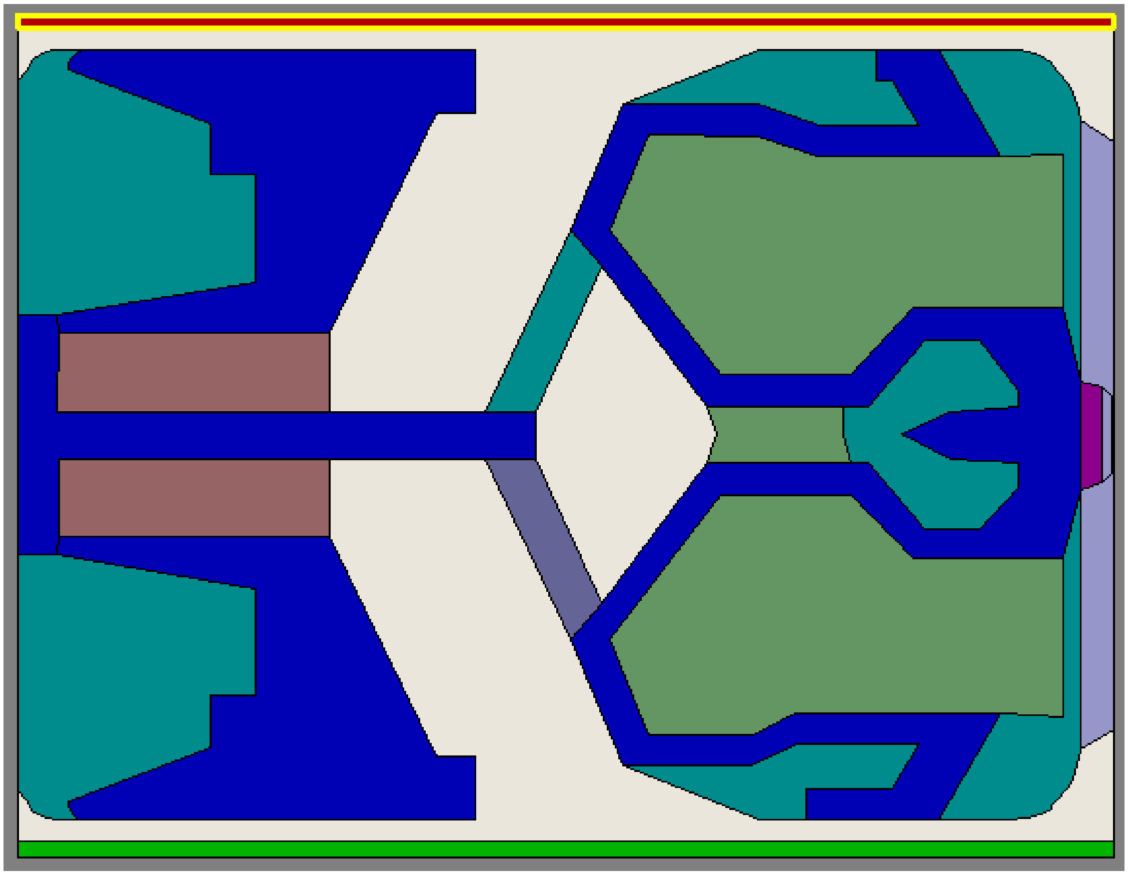


Figure 10. Slider 8 rails

Quick Solver type	Tesla K20 GPU	Quick 4	Speed increase
Execution time	1 minute	2 minutes	50 %

Table 10 a. Tesla K20 GPU solver vs Quick 4 solver

Quick Solver type	GeForce 680 GPU	Quick 4	Speed increase
Execution time	1 minute	2 minutes	50 %

Table 10 b. GeForce GPU solver vs Quick 4 solver

Discussion and conclusions:

The figures and results are essentially self-explanatory. As seen for the first slider, the reason why there was no speed improvement or 0 % speed increase is because the slider is not a complex design. This means that both the CPU and GPU solvers solve the problem so fast that you cannot see any difference in execution time. While for other more complex sliders, such as slider 3 and 5, both sliders have many rails and different etching levels, which causes the high speed improvement of 66 % and 60 %, respectively, when running the case on a GPU. For that reason, the GPU is recommended for complex slider designs, which are the common case in current hard disk drives.

With some of the complex slider designs, there is a noticeable difference in Quick 4 computation times between tables a and b. For example, looking at table 9.a and 9.b, the Quick 4 solver took 5 minutes in table 9.a while only 2 minutes in table 9.b. This difference can be explained by noting the different desktop computers used to run the Tesla K20 and GeForce 680 GPUs. The Tesla K20 was installed on a slower desktop tower with 2.2 GHz CPU clock as compared to the 3.6 GHz Intel i7 that was on the Dell XPS8500 used with the GeForce 680.

If you have not decided to buy a relatively expensive ~$3,600 Tesla K20 GPU along with minimum of $1,500 high power tower desktop with multiple PCI express slots, our recommendation is to buy the more affordable, $450 GeForce 680 GPU. It does the job well and has closely comparable results with the Tesla K20, minus the higher price.

In conclusion, as we see in the example cases along with their output results, the GPU Quick 5.0 is a significantly faster solver for complex modern slider designs.