`Aperture4D`: GPU Dynamic Aperture Evaluator Manual

Stephen Brooks

June 21, 2011

1 Introduction

This program uses fragment shaders (also called pixel shaders) on the computer graphics card (GPU) to rapidly track thin lens lattices for millions, billions or trillions of different starting points in 4D phase space. It outputs the volumes of 4D phase space that remain within the aperture (specified by collimators) after different numbers of cells.

1.1 Installation and Running

Copy the directory (at least `aperture4d.exe`, `font.dat` and a valid `input.txt`) to your local machine to run. Output will be written to `output.csv`.

Requires Windows and graphics hardware (GPU) capable of at least OpenGL 2.0 (the code will tell you if not). It likely will work under Linux using WINE and Mac OS X using the WINE component of WineBottler.

Press Q or Escape, or close the window to quit (S takes a screenshot).

2 Units and Conventions

This program is relatively unit-agnostic, hence the lack of commands for specifying particle type. It is assumed that units are chosen such that a drift of length l will have $x_{\text{out}} = x_{\text{in}} + lx'_{\text{in}}$ and a quadrupole kick of strength k will focus in the x plane by $\Delta x' = -kx$. Currently this means it is appropriate only for a single momentum of particle. Higher multipoles by convention kick particles on the positive x axis towards negative x via $\Delta x' = -sx^n/n!$, where s is the multipole strength and $n = 2$ for sextupole, $n = 3$ for octupole and so on.

3 Input File (`input.txt`) Example

As a quick start, a very simple `Aperture4D` input file looks like the following.

```
xrange -4 4
xprange -2 2
yrange -4 4
yprange -2 2
xcollimator -3.162 3.162
# ycollimator -1e20 1e20 # no y collimator uses "large" limits
iterations 10 100 1000 2000 5000
# Cell lattice follows
D 1
K 1
D 1
K -1 -0.2
```

This scans the phase space hypercuboid with $x, y \in [-4, 4]$ and $x', y' \in [-2, 2]$ and a collimator set to exclude particles with $|x| > 3.162$. The y collimator is not used and is by default set to a very large value. Hashes (`#`) begin comments. The simulation will run for 5000 cells with statistics on particle survival output after 10, 100, 1000, 2000 and 5000 cells.

The lattice in this cell is very simple: an O-F-O-DS arrangment where the F is a focussing quadrupole and the DS is a defocussing quadrupole combined with a sextupole of strength -0.2. Drifts are 1 metre (or other consistent unit) long and quadrupoles are of unit strength.

4 Input File: Basic Parameters

4.1 {x,xp,y,yp}range

It is advised to specify the ranges of the phase space scan in all four dimensions. Each command `xrange`, `xprange` etc. takes two parameters: the minimum and maximum value of the range. So a complete hypercuboid can be specified by four lines similar to the ones below.

```
xrange -0.1 0.1
xprange -0.02 0.02
yrange -0.1 0.1
yprange -0.01 0.01
```

4.2 {x,y}collimator

Particles are lost when their x or y coordinates fall outside the ranges specified by the collimator component at the end of a cell. `xcollimator` takes two parameters: the minimum and maximum allowed x coordinate for particles at the end of the cell. `ycollimator` does the same for y. It is not necessary to specify both collimators, but if neither are specified the code will allow a particle coordinates to become very large before they are lost, so specifying at least one is advised, as below.

```
xcollimator -0.03 0.04
```

4.3 `iterations`

This command takes a list of one or more numbers of cells the particles should be tracked through, in ascending order, e.g.
`iterations 10 100 250 1000`.
For each starting phase space position, the particles will be tracked for the maximum number of cells (1000 here) but statistics on survival will be output at all the specified values.

5 Input File: Lattice Elements

5.1 `D` (drift)

This element takes one parameter: the drift length. So the line
`D 0.5`
will create a drift of length 0.5.

5.2 `K` (quadrupole or multipole kick)

This element takes one or more parameters specifying the strengths of multipoles in the kick, starting from quadrupole, i.e.
`K` *quad-strength sext-strength oct-strength ...* .
Earlier values may be left at zero for pure higher multipoles. As the kick has no length, the kicks from different multipoles are added together linearly.

6 Input File: Advanced Parameters

6.1 `baseres` (base resolution)

This is the number of sample points along each axis of the 4D scan grid that is plotted on each 'frame'. By default this is set to 32, so each frame consists of $32^4 = 1048576$ points arranged in a 1024×1024 grid ($1024 = 32^2$). Choosing a lower value, e.g. 10 with
`baseres 10`
will use a coarser grid, so frames are calculated more rapidly, which is helpful for simulations with a large number of iterations. This also means output data is produced at lower and more intermediate scan resolutions.

7 Output File (`output.txt`)

Output from `Aperture4D` will be produced whenever the number of samples (entire screens plotted) is n^4 for some whole number $n \geq 1$. The graphical display will tell you how many samples until the next output point is reached. The output is comma-separated and is a sequence of tables, summarising the results from successively finer sampling grids of the phase space.

```
Aperture4D job started 2011-Jun-21 15:15:01
32^4 grid
Iters,4D_Vol
Domain,1024
10,181.121094
100,91.4150391
1000,69.4453125
2000,64.2978516
5000,57.9248047

64^4 grid
Iters,4D_Vol,+/-
Domain,1024,0
10,181.49054,0.0737877747
100,91.2980957,0.0413887131
1000,69.3467407,0.0357050656
2000,64.0964966,0.0304342245
5000,57.8505249,0.0502938749
```

Here, each table starts with the sampling resolution. For a 'base resolution' B, where $B = 32$ by default but can be changed with the `baseres` command, output is given for grids of resolution $(nB)^4$ (i.e. $nB \times nB \times nB \times nB$) for $n \geq 1$.

The columns are: the number of iterations (cells simulated), the estimated volume of 4D phase space still transmitted at that point and a 1σ error estimate for that volume (not given in the first table because of insufficient statistics). 4D phase space volumes are measured in a unit such as $(\text{mm.mrad})^2$ or whatever the unit of $xx'yy'$ is in your input. For SI units, this would be the rather large $(\text{m.rad})^2$. The `Domain` row gives the total 4D volume being scanned, for reference.

8 Benchmark

On a test problem, the GPU could calculate a batch of 1048576 starting points 4.3 times per second, whereas a single CPU core running the same code averaged 0.193 batches per second. The CPU was an Intel Xeon X5355 (2.66GHz, 4 cores) and the GPU was an ATI Radeon HD5670 (790MHz, 400 unified shader cores). This GPU would therefore be 5.6× faster than the CPU even if it had used all four cores.