To reduce the gDPM output files while keeping the needed for our project information, the following changes were made in the “rungdpm.cu” file of the gDPM toolkit[1,2]:

```
/*#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
#include <thrust/sort.h>
#include <thrust/copy.h>*/
#include <thrust/device_vector.h>
#include <thrust/remove.h>
#include <algorithm>
```

```
// Sams addition - Make predicate easy to write
typedef thrust::tuple< float4, float4 > IntTuple2;

struct isntZero
{
    __host__ __device__
    bool operator()(const IntTuple2& tup)
    {
        float xdirmin = 0.965f;
        //float xphantommax = 35.0f;
        //const float4 po = thrust::get<0>( tup );
        const float4 di = thrust::get<1>( tup );
        return xdirmin >= di.x; //|| xphantommax >= po.x;
    }
};
```

```
311                 // PSF by Sam ************************************************************
312                 thrust::device_vector<float4> posvec(PSFx, PSFx + nactive_h);
313                 thrust::device_vector<float4> dirvec(PSFvx, PSFvx + nactive_h);
314
315                 // Make zip_iterator easy to use
316                 typedef thrust::device_vector< float4 >::iterator  IntDIter;
317                 typedef thrust::tuple< IntDIter, IntDIter >     IntDIterTuple2;
318                 typedef thrust::zip_iterator< IntDIterTuple2 >  ZipDIter;
319
320                 // Remove elements in many vectors if element in vec0 is negative
321                 ZipDIter newEnd = thrust::remove_if(    thrust::make_zip_iterator( thrust::make_tuple( posvec.begin(), dirvec.begin() ) ),
322                 thrust::make_zip_iterator( thrust::make_tuple( posvec.end(), dirvec.end() ) ),isntZero() );
323
324                 // Erase the removed elements from the vectors
325                 IntDIterTuple2 endTuple = newEnd.get_iterator_tuple();
326                 posvec.erase( thrust::get<0>( endTuple ), posvec.end() );
327                 dirvec.erase( thrust::get<1>( endTuple ), dirvec.end() );
328                 //Reducing vector size
329                 int newSize = posvec.end() - posvec.begin();
330                 std::cout << "New size: " << newSize << std::endl;
331                 float4 posout[newSize];
332                 float4 dirout[newSize];
333                 thrust::copy(posvec.begin(), posvec.end(), posout);
334                 thrust::copy(dirvec.begin(), dirvec.end(), dirout);
335
336                 // Write output pos array to file
337                 std::ofstream outputFile("./output/posvec.txt",std::ios::app);
338                 for (int i = 0; i < newSize; ++i)
339                 {
340                     outputFile << posout[i].x << " " << posout[i].y << " " << posout[i].z << " " << posout[i].w << "\n";
341                 }
342                 outputFile.close();
343
344                 // Write output dir array to file
345                 std::ofstream outputFile1("./output/dirvec.txt",std::ios::app);
346                 for (int i = 0; i < newSize; ++i)
347                 {
348                     outputFile1 << dirout[i].x << " " << dirout[i].y << " " << dirout[i].z << " " << dirout[i].w << "\n";
349                 }
350                 outputFile1.close();
351                 // end ******************************************************
```

References:

1. Jia, X. & Jiang, S.B. (2011). gDPM v2.0. A GPU-based Monte Carlo simulation package for radiotherapy dose calculation. The Center for Advanced Radiotherapy Technologies (CART), UCSD.
2. Jia, X., Ziegenhein, P., & Jiang, S. B. (2014). GPU-based high-performance computing for radiation therapy. Physics in Medicine and Biology, 59(4), p. R151–R182.