POSTER P5190 | CONTACT NAME Carlos Gonzalez: cagc@fisica.unam.mx

Modeling X-Ray Diffractometer Device Function Using MonteCarlo Ray Tracing and CUDA

Xim Bokhimi and Carlos González

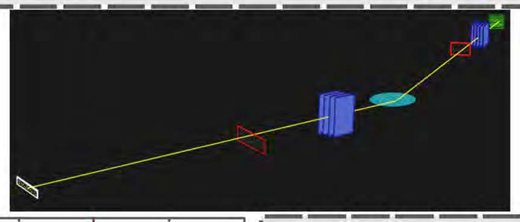
UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

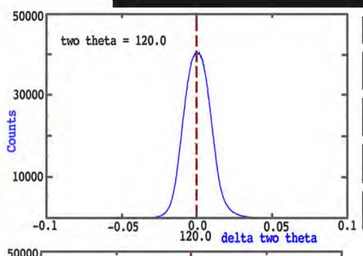
if 75 años INSTITUTO DE FISICA

Ejecution Flow

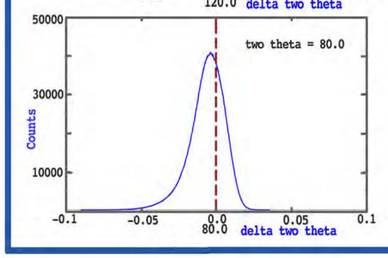
```
class  input_geometry {
  geometry_type(reflexion,
        transmision, capillary)
  GoniometerRadius(radius)
  X-Ray_Source(Width,Length)
  DivergenceSlit(
        DistanceToAxialAxis,W,L)
  PrimarySoller(angle | A,B)
  Sample(Diameter,Thickness | W,L)
  AntiscatteringSlit(
        DistanceToAxialAxis,W,L)
  SecondarySoller(angle | A,B)
  Detector(Width, Length,
        ArrayWidth)
  Monocromator(
        Spectrometer Radius |
        Monocromator Angle)
}
```

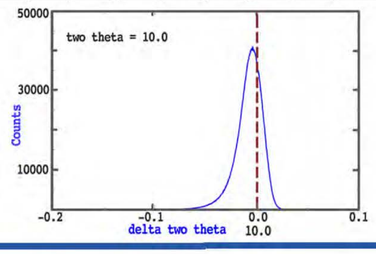

```
cuRand: pseudorandom numbers of the Mersenne Twister  algorithm.

foreach q in (N-angles)  {
  i = blockIdx * blockDim + threadIdx;
  RayIni[i] = cuRandMakeRay(goniomRand, DivSlitRand, PrimSoller, ...)
  if( RayIni[i] == SUCCESS)
    RayOut[i] = cuRandMakeRay(AntiScateringSlit,SecSoller,Detector,...)
    if(RayOut[i] == SUCCESS)  {
      makeBragg(RayIni[i],RayOut[i], deltaTwoTheta[i])
      Intensity[deltaTwoTheta[i]]++
  SaveIntensityFile(deltaTwoTheta[i], Intensity[i])
}
```






```
class  printGraphs {
  setPipe(gnuplot | canvas | OpenGL)
  setSettings(...)
  plot( x = two theta,
        y = Counts, ...)
  freeAll(all Memory...)
}
```





The quantitative analysis of X-ray diffraction patterns provides information about the atom distribution of the different phases in a sample that helps to understand its macroscopic properties. This analysis can be done by refining the crystalline structure of the phases with the Rietveld method. In general, most of the X-ray diffraction experiments are performed in a standard laboratory. Therefore, to obtain confident information about the phases in the sample from this analysis, it is necessary to have a good model for the experimental arrangement, in order to separate its contribution to the diffraction pattern, which produces aberrations, from the contribution of the sample. Therefore, there is a need to have open source codes that take this fact into account. In this work we present the advance we have about the development of the code BORO to refine crystalline structures using the Rietveld method. Until now, we have a first final version for the part of the code that models the experimental arrangement, which was done using the ray-tracing method. The software was written, for the Qt platform, in C++, OpenGL and CUDA for CPU clusters and GPUs; it runs in Linux and OS X. In the plan for the development of the BORO code, it will be connected with software that models the atomic distributions of small and large atomic clusters, amorphous or crystalline, and with software that models diffraction patterns using the Debye Function.

Geometry Construction

The optical elements of the diffractometer must be defined to calculate the device profile function.
Optical geometry (Bragg-Brentano, Debye-Scherrer, transmission).
The following parameters correspond to a theta-theta Bragg-Brentano reflection geometry.
Goniometer radius.
Divergence slit: width, length, and position relative to axial axis.
Primary Soller: distance between plates and their length, or, its axial divergence angle.
Sample: equatorial and axial dimensions, or, diameter.
Antiscattering slit: width, length, and position relative to axial axis.
Detector slit: width and length.
Detector dimensions: zero, one or two dimensional.
Monochromator (if it is present), spectrometer diameter, monochromator angle.

Random Events

For each theta angle, a large number of random events are computed in parallel as follows:
A random position is generated at the area of the X-ray source
A random position is generated at the divergence slit along the with dimension.
A random position is generated at the primary Soller along the axial axis.
With these random positions a ray path is generated.
The set of the rays reaching the sample define the true area.
Then, random position is generated at the area defined by the detector slit.
If the detector is one dimensional, a random position is also generated along the one-dimension.
A random position is generated at the secondary Soller along the axial axis.
A random position is generated in the area defined by the antiscattering slit.
With all these positions in the secondary optic a ray is traced.
If it falls on the true area of the sample, a true event is defined.

Bragg Condition

For each true event, the Bragg angle 2θ, defined between the primary and secondary ray paths, is computed.
The randomness of the events generates a dispersion of 2θ angles, which depends on the theta angle of interest and the dimensions of the optical components.
After having reached a predetermined number of true events, or the maximum number of rays, the random process comes to end.

Device Profile

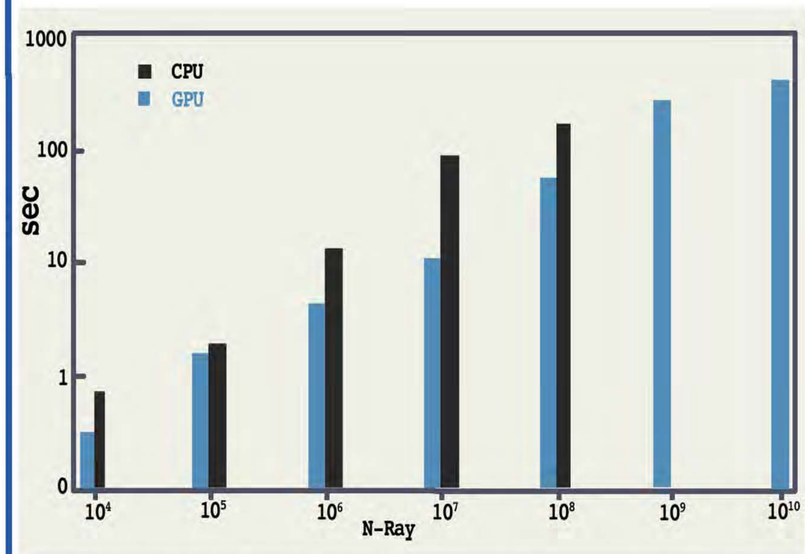
Finally, we have a representation of the device profile as a collection of noisy points. This give an effective and low parametric representation of the samples difraction function.

References

[1] Lablonski, Jagodzinski and Pajek., Phys. Scr. T156 (2013) pp.1-3
[2] G. J. Ward, F. M. Rubinstein and R. D. Clear., SIGGRAPH Comput. Graph. Vol.22, (1998) pp.85-92
[3] J. Bergmann and R. Kleeberg,, J.Appl. Cryst. vol.34, (2001) pp. 16-19
[4] Young and Wiles,, J. Appl. Cryst., 15 (1982) pp. 430-438
[5] Aila, T. and Laine S. In Proceedings of High-Performance Graphics. (2009), pp145-149

Preliminary Results

High Productivity:
The software provides a concise and compact algorithm description.
Portability:
It is supported in a large variety of architectures including the newest NVIDIA GPU.
Competitive Perfomance:
At least ten times faster than the algorithm in CPU. The number of Rays that can be generated depends on the GPU.



We tested our code using Nvidia GT Force 755M, compared to the sequential implementation running on a intel i7 CPU at 2.4 GHz. The speed increase is up to10 times, depending on the GPU clock and its memory. For example, the GPU Tesla K20 was 20% faster than the GT Force GPU, and can generate more rays.

Future Work

Future tuning of the ported application for better usage of the GPU capability and adaptation to different multi-GPUs environments.

Improvement the kernels in the GPU.

Dividing parallel tasks amount the number and type of GPUs.

We are working on a better characterization of the optical geometry of the diffractometer.