# Inverse Modeling Nanostructures from X-Ray Scattering Data through Massive Parallelism

[Extended Abstract]

Abhinav Sarje
Computational Research Division
Lawrence Berkeley National Laboratory
asarje@lbl.gov

Dinesh Kumar
Advanced Light Source
Lawrence Berkeley National Laboratory
dkumar@lbl.gov

Singanallur Venkatakrishnan
Advanced Light Source
Lawrence Berkeley National Laboratory
svvenkatakrishnan@lbl.gov

Slim Chourou
Computational Research Division
Lawrence Berkeley National Laboratory
stchourou@lbl.gov

Xiaoye S. Li
Computational Research Division
Lawrence Berkeley National Laboratory
xsli@lbl.gov

Alexander Hexemer
Advanced Light Source
Lawrence Berkeley National Laboratory
ahexemer@lbl.gov

## 1. INTRODUCTION

We consider the problem of reconstructing material nanostructures from grazing-incidence small-angle X-ray scattering (GISAXS) data obtained through experiments at synchrotron light-sources. The GISAXS technique is a key method to probe material structures and morphologies at the nanoscale and gather valuable information about the organization, disorder and other structural parameters. This is useful in a number of applications such as design and fabrication of high-density energy storage devices, organic photovoltaics, and drug design. Analysis of experimentally collected scattering data has been the primary bottleneck and a challenge in this process due to lack of efficient computational methods, augmented by the high data throughput of current state-of-the-art detectors at beamlines.

The inverse modeling process for nanostructure reconstruction is essentially a "parameter optimization" problem. In a basic description of an iterative solver for this problem, an input structural model, starting possibly with an initial guess of parameter values, is modified and refined iteratively based on an error measuring the distance between the experimentally generated scattering pattern and the simulated pattern for this model. The updates and refinements of the model are carried out until a convergence is reached and the corresponding converged structural model is taken as the generated solution.

## 2. SOLUTION

We exploit the availability of massive parallelism in leadership-class supercomputers with multi-core and graphics processors to realize the compute-intensive reconstruction process. To develop a solution, we employ various optimization algorithms including gradient-based LMVM, derivative-free trust-region-based POUNDerS [3], and particle swarm optimization [2].

The objective function in this inverse modeling process is a forward simulator [1] which computes the X-ray scattering patterns for a given sample structure model, and then compares it with the experimental data producing a Chi-square error measure. This forward simulation is based on the Distorted Wave Born Approximation technique, and is the most compute-intensive component. We have developed a parallel high-performance simulation code [4] which is able to take advantage of available parallelism and can run efficiently on multiple nodes of multi-core CPUs as well as GPUs. It is implemented using MPI along with OpenMP and Nvidia CUDA.

We utilize the high-performance implementations of LMVM and POUNDerS optimization algorithms available in the TAO software package [3]. Starting with an initial guess, in each iteration both the LMVM and POUNDerS algorithms perform one evaluation of the objective function, hence, the total time to solution is linearly proportional to the number of iterations, and hence, the number of objective function evaluations, performed for convergence. We run experiments with simulated data to evaluate the convergence of these methods for a 2-parameter and a 6-parameter optimization. In the former case starting with varying initial guesses, both LMVM and POUNDerS were able to converge, with LMVM requiring on average 3 times the number of iterations than POUNDerS. In the latter case, LMVM failed to converge, and POUNDerS showed a convergence rate of less than 30%.

We further develop a parallel implementation of the particle

swarm optimization (PSO) method. This method consists of a set of $n$ agents, each of which moves along a different path within the parameter search space exploring the parameter values along the way and converging towards the optimal solutions. The path of an agent is determined stochastically and is influenced relatively by its current best found parameter value set along its traveled path, and the globally best found values among all agents' best value sets. The initial positions of the agents is determined randomly and hence do not require an initial guess. Since each agent can evaluate the objective function independently of all other agents, our implementation is able to exploit large-scale parallelism effectively. The available compute resources in a system are evenly divided among all the agents in a run, and each agent performs simulations in parallel using its allocated resources. A global reduction operation per iteration is required to identify the global best solution. Hence, when all agents are able to run simultaneously, the time to solution is linearly proportional to the number of iterations required for convergence.

To evaluate the convergence performance of our PSO implementation, we run experiments on the 2 and 6 parameter optimization cases and observe that PSO has nearly 100% convergence rate, and is hence more robust than LMVM and POUNDerS methods for the application at hand. Furthermore, a run with larger number of agents requires lesser number of iterations to converge than a run with smaller number of agents. For example, a run with 100 agents took 15 iterations for convergence while a run with 25 agents took about 40 iterations. This reaffirms our case of the use of large-scale parallel systems. We also demonstrate the computational speed on the Titan supercomputer with real data obtained through X-ray scattering experiments on organic photovoltaic samples. The time taken on 500 GPU nodes with 20 agents and 20 iterations (400 evaluations) was 3110 seconds, on 2000 GPU nodes with 50 agents and 20 iterations (1000 evaluations) was 2071 seconds, and on 8000 GPU nodes with 80 agents and 20 iterations (1600 evaluations) was 865 seconds, delivering near linear scaling.

# 3. REFERENCES


[1] S. Chourou, A. Sarje, X. Li, E. Chan, and A. Hexemer. HipGISAXS: A High Performance Computing Code for Simulating Grazing Incidence X-Ray Scattering Data. *Journal of Applied Crystallography*, 46(6):1781–1795, 2013.

[2] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, pages 1942–1948. IEEE, 1995.

[3] T. Munson, J. Sarich, S. Wild, S. Benson, and L. C. McInnes. Tao 2.0 users manual. Technical Report ANL/MCS-TM-322, Mathematics and Computer Science Division, Argonne National Laboratory, 2012. http://www.mcs.anl.gov/tao.

[4] A. Sarje, X. Li, S. Chourou, E. Chan, and A. Hexemer. Massively Parallel X-ray Scattering Simulations. In *International Conference for High Performance Computing, Networking, Storage and Analysis (Supercomputing)*, 2012.