How fast is DDA? A reproducible cross-code comparison

C. Argentin[1], P. C. Chaumet[2], M. Gross[3], and M. A. Yurkin[1]

[1] *Université Rouen Normandie, INSA Rouen Normandie, CNRS, CORIA UMR 6614, Rouen, 76000, France*
[2] *Institut Fresnel, Aix Marseille Univ, CNRS, Centrale Marseille, Marseille, 13013, France*
[3] *Université Montpellier II, CNRS, L2C UMR 5221, Montpellier, 34090, France*

We present a fair CPU and GPU benchmark comparison of the discrete dipole approximation (DDA) codes `DDSCAT`, `ADDA`, and `IFDDA` by strictly aligning all numerical parameters to achieve machine-precision agreement between implementations. The results presented here are derived from our recent study and identify the practical strengths and limitations of each implementation, provide concrete best practices for users and developers, and highlight directions for improving the performance and usability of modern DDA codes.

1 Introduction

Light scattering by particles of arbitrary shape plays a central role in many areas of physics, including atmospheric science and nanophotonics. Since analytical solutions of Maxwell's equations exist only for a limited set of geometries, numerical methods are required for realistic modeling. Among these methods, the discrete dipole approximation (DDA) [1] has become one of the most widely used approaches due to its simplicity, and the availability of mature open-source implementations.

The most widely used open-source DDA codes today are `DDSCAT` [2], `ADDA` [3], and `IFDDA` [4]. Although these codes share the same physical foundation, they differ in their linear-system

formulations, FFT implementations, parallelization strategies, and free-parameter choices. As a consequence, quantitative comparisons between codes are difficult to interpret [5] because the implementations may exhibit different accuracy and computational performance.

The objective of this work is therefore twofold. First, we establish a unified and reproducible methodology that enables floating-point–consistent cross-verification between independent DDA implementations. Second, using this baseline, we perform fair CPU and GPU performance comparisons across modern hardware platforms. For this, we introduce a Python framework, `dda-bench`, which automates cross-code comparisons and quantifies agreement in terms of matching significant digits.

2 Methodology

Although the DDA is a deterministic method, independent implementations may produce different numerical results. In this work we distinguish between three practical levels of agreement. Machine-precision agreement corresponds to roughly 10–15 matching digits in double precision and indicates that two implementations are solving numerically equivalent linear systems. A second regime, referred to as η-level agreement, occurs when results differ only within the solver tolerance. Finally, method-error agreement arises when discrepancies are dominated by discretization or formulation differences. Only the first regime enables truly fair performance benchmarking.

3 Configuration

All benchmarks were performed on a single well-defined test case consisting of a homogeneous ice cube with size parameter $kD = 30$. The simulations use the filtered coupled dipoles (FCD) formulation with the BiCGStab iterative solver. Particular care was taken to tune the stopping thresholds so that all implementations terminate after the same number of iterations,

thereby eliminating solver-history bias.

The CPU benchmarks were conducted on two platforms representative of both high-performance computing and everyday usage: a dual-socket AMD EPYC 9654 node and an Intel Core Ultra 7 165H laptop processor. GPU benchmarks were performed on four NVIDIA accelerators spanning workstation and datacenter hardwares, namely the RTX 2000 Ada, RTX 6000 Ada, A100, and H200. Each simulation was repeated three times and averaged in order to reduce run-to-run variability.

4 CPU performance

Our results confirm that FFT backend choice is critical because the dominant cost of DDA simulations is the FFT-accelerated matrix–vector product, which directly impacts achievable problem sizes and computation times. Implementations relying on FFTW and Intel MKL consistently enable shorter runtimes. Across both tested CPU platforms, `ADDA` and `IFDDA` generally provide the shortest execution times, while `DDSCAT` compiled with MKL becomes competitive on Intel architectures.

When multiple cores are used, the practical benefit depends on how efficiently memory bandwidth and parallel resources are exploited. `ADDA` shows the strongest scaling behavior using its MPI implementation, which also allows users to use several nodes. `IFDDA`, with its OpenMP implementation scales efficiently up to the point where memory bandwidth becomes the limiting factor which typically occurs earlier than for `ADDA`. Practically, for a grid size of $n_x = 250$, `ADDA` outpeforms `IFDDA` by a factor or 3 on a cluster node using 125 CPU cores. In the present configuration, `DDSCAT` shows limited speedup due to the lack of OpenMP parallelization in its main FFT routines.

From a memory standpoint, switching `DDSCAT` to single precision can reduce the memory footprint by approximately a factor of two, which may enable larger simulations on memory-constrained systems while still preserving η-level accuracy for well-conditioned problems. This option can therefore be useful for studies requiring parameter sweeps.

5 GPU performance

On modern accelerators, the main question is how effectively GPU memory and bandwidth can be leveraged to accelerate DDA simulations. `IFDDA` performs the entire iteration process on the GPU, thereby minimizing host–device transfers and making efficient use of available memory bandwidth. In contrast, the default OpenCL mode of `ADDA` offloads only the FFT operations while keeping the iterative solver on the CPU. Our results show that, this hybrid workflow introduces additional data movement that limits the effective acceleration on recent GPUs.

Across the tested hardwares, `IFDDA` consistently shows the shortest runtimes, providing speedup up to 2× over `ADDA` and slightly more in single precision The experimental `OCL_BLAS` mode of `ADDA` narrows this gap, demonstrating that full solver offloading is a promising direction for future development.

As mentioned, GPU memory capacity remains the primary limiting factor for very large DDA simulations. Practically, for a grid size $n_x = 250$, `IFDDA` require 13.4 and 28.5 GiB of GPU memory in single and double precision, respectively. In contrast, `ADDA` requires 6.5 GiB for the same grid size, which allows it to run larger simulations on memory-limited devices.

6 Conclusion

We have presented a methodology for floating-point–consistent cross-verification and fair benchmarking of the three major open-source DDA solvers: `DDSCAT`, `ADDA`, and `IFDDA`. Using the `dda-bench` framework, we demonstrated that machine-precision agreement between independent implementations is achievable when all physical and numerical parameters are harmonized. This controlled baseline enables meaningful performance comparisons that isolate genuine algorithmic and architectural effects.

Our benchmarks show that the FFT implementation is the primary driver of performance, that `ADDA` provides the strongest CPU scalability, and that fully GPU-resident solvers such as

`IFDDA` deliver the largest acceleration on modern accelerators. These results provide practical guidelines for users willing to optimize their DDA simulations, and suggest clear directions for further code optimization, particularly in the area of GPU offloading strategies. More details can be found in Ref. [6].

References

[1] M. A. Yurkin, *Discrete dipole approximation*, in Light, Plasmonics and Particles (Elsevier, 2023), M. P. Menguc and M. Francoeur, eds., pp. 167–198.

[2] B. T. Draine, P. J. Flatau, *Discrete dipole approximation for scattering calculations*, J. Opt. Soc. Am. A **11** (1994), 1491–1499.

[3] M. A. Yurkin, A. G. Hoekstra, *The discrete dipole approximation code ADDA: Capabilities and known limitations*, J. Quant. Spectrosc. Radiat. Transfer **112** (2011), 2234–2247.

[4] P. Chaumet, D. Sentenac, G. Maire, M. Rasedujjaman, T. Zhang, A. Sentenac, *IFDDA, an easy-to-use code for simulating the field scattered by 3D inhomogeneous objects in a stratified medium: tutorial*, J. Opt. Soc. Am. A **38** (2021), 1841–1852.

[5] A. Penttilä, E. Zubko, K. Lumme, K. Muinonen, M. A. Yurkin, B. T. Draine, J. Rahola, A. G. Hoekstra, Y. Shkuratov, *Comparison between discrete dipole implementations and exact techniques*, J. Quant. Spectrosc. Radiat. Transfer **106** (2007), 417–436.

[6] C. Argentin, P. C. Chaumet, M. Gross, M. A. Yurkin, *Floating-point–consistent cross-verification methodology for reproducible and interoperable DDA solvers with fair benchmarking*, arXiv **2603.02871** (2026).