Giving Second Life to a Scientific Software: Introducing Hydrogen Bond Analysis Tool (HBAT) v2

Abhishek Tiwari

Citation: *A. Tiwari*, "Giving Second Life to a Scientific Software: Introducing Hydrogen Bond Analysis Tool (HBAT) v2", Abhishek Tiwari, 2025. doi:10.59350/j7243-tf172

Published on: June 30, 2025

Every so often, we get a chance to look back at our past work. Sometimes it's with a cringe, sometimes with nostalgia, and occasionally, with an opportunity to rebuild. During a recent and much-needed time off, I found myself embarking on a project that was a blend of all three: a complete, ground-up rewrite of the Hydrogen Bond Analysis Tool (HBAT), a piece of software I first wrote nearly two decades ago.

Today, I am thrilled to announce the result of that effort: HBAT v2. It's a modern, open-source, and extensible tool built in Python. What began as a humble PERL script during my internship at the *University of Hyderabad* in 2005 has now been given second life as a modern, open-source Python application that will continue to serve the structural biology and drug discovery community.

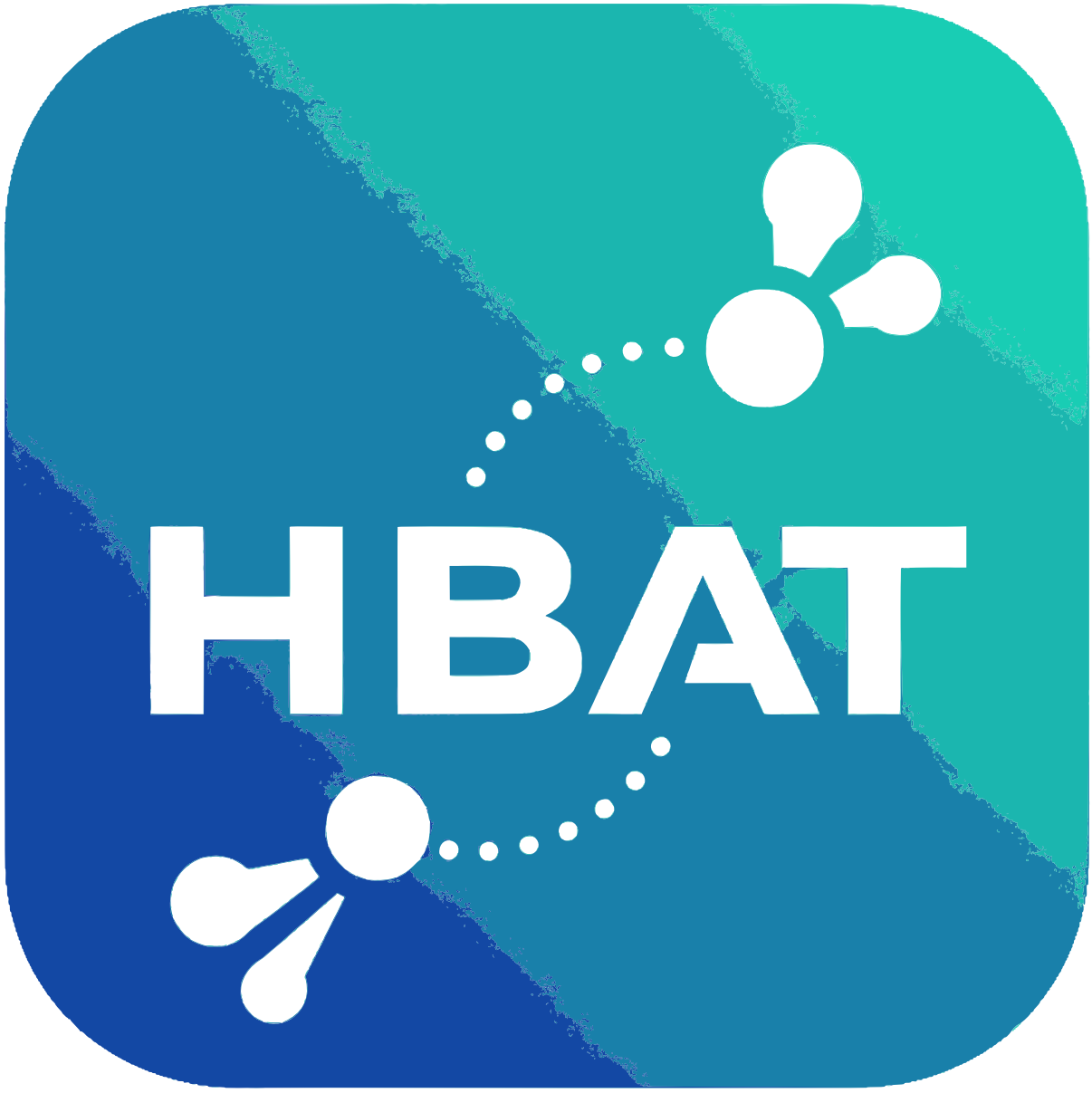

Figure 1: Hydrogen Bond Analysis Tool (HBAT) v2

The Genesis of HBAT

The original HBAT emerged from a practical need during my internship at the *University of Hyderabad* circa 2005. The resulting PERL script, though modest in its initial form, addressed a genuine gap in the scientific community. When the HBAT was published in the journal [1], I hoped it might prove useful to other researchers facing similar challenges. Little did I anticipate the impact it would have.

A Tool That Found Its Community

Since making the HBAT (and its sister utility [2]) available on SourceForge in 2007, the software has been downloaded more than 11,650 times, demonstrating sustained interest from the research community. Original paper covering the HBAT has garnered over 77 citations, indicating that the tool has contributed to subsequent research and discoveries. HBAT has been cited by more than 135 researchers across 28 different countries, truly making it a global scientific resource. Researchers from prestigious institutions including the Wellcome Centre for Cell Biology, National Cancer Institute, Johns Hopkins University, Mount Sinai, and the Indian Institute of Science have incorporated HBAT into their research workflows.

The Challenge of Legacy Code

However, success brought its own challenges. The original HBAT served the community well, offering a PERL/TK-based graphical user interface with Windows binaries. Over the years, I regularly received requests for Mac and Linux binaries, as well as suggestions for new features and improvements. The enthusiasm of the user community was both inspiring and frustrating, as I found myself unable to adequately respond to these reasonable requests.

The root of the problem lay in two fundamental issues with the original implementation. First, the codebase was what I can only describe as “scrappy”. It was written quickly to solve an immediate problem without the benefit of modern software engineering practices. The code lacked adequate testing, proper documentation, and well-defined interfaces. This made extending functionality a cumbersome and error-prone process.

Second, the entire application was implemented as a single PERL script. I know what you’re thinking, and yes, in retrospect, this was less than ideal. While this approach had the advantage of simplicity for initial development, it created significant technical debt that compounded over time. Adding features meant navigating an increasingly complex monolithic codebase, and ensuring cross-platform compatibility became a Sisyphean task.

These technical limitations meant that despite the community’s enthusiasm and my desire to improve the tool, progress stagnated. The original HBAT remained functional but frozen in time, unable to evolve with the changing needs of its users or take advantage of advances in programming languages and libraries.

The Catalyst for Change

Two factors aligned to make HBAT v2 possible: extended time off and the availability of Claude Code. The luxury of uninterrupted time allowed for the deep focus required to undertake a complete rewrite,

while Claude Code provided invaluable assistance in navigating the complexities of porting legacy PERL code to modern Python.

The decision to rewrite rather than incrementally improve was deliberate. While refactoring might have been faster in the short term, it would have perpetuated the fundamental architectural problems that limited the original tool. A complete rewrite offered the opportunity to apply nearly two decades of accumulated wisdom about software engineering, scientific computing, and user experience.

Claude Code proved instrumental in making this ambitious rewrite feasible. As an agentic command-line tool, Claude Code transformed what could have been months of tedious porting work into an efficient, collaborative process. In short, I am completely blown away by Claude Code. What might have taken a solo developer 6-12 months to complete was accomplished in weeks, with higher code quality than I could have achieved working alone.

Python emerged as the natural choice for the new implementation. Its rich ecosystem of scientific libraries, cross-platform compatibility, and readable syntax make it ideal for scientific software development. The language's emphasis on clarity and simplicity aligns well with the goal of creating maintainable, extensible code that can evolve with user needs. The move to Python finally solves the long-standing request for multi-platform support. The HBAT v2 runs natively on Windows, macOS, and Linux.

HBAT v2: Modern Science, Modern Code

The HBAT v2 represents a fundamental reimagining of the tool while preserving the scientific functionality that made the original version valuable. The new implementation embraces modern software engineering practices, starting with a clean, modular architecture that separates concerns and provides well-defined interfaces between components.

One of the most significant improvements is the dual interface approach. The HBAT v2 provides both a graphical user interface (GUI) built with tkinter and comprehensive command-line interface (CLI). This flexibility recognizes that different users have different preferences and workflow requirements. Researchers who prefer point-and-click interfaces can use the GUI, while those integrating HBAT into automated pipelines can leverage the command-line tools.

The rich API represents perhaps the most forward-looking aspect of the HBAT v2. Rather than treating the tool as a black box, the new architecture exposes core functionality through well-documented programming interfaces. This allows advanced users to incorporate the HBAT v2 functionality into larger applications, create custom analysis workflows, or extend the tool's capabilities to address specific research questions (see API example).

To get started, run `pip install hbat` and you are all set to use `hbat` as CLI or launch the HBAT

GUI using `hbat-gui`. In addition, an installer for macOS is available now with Windows and Linux installers planned in next release.

Open Source, Open Science

A crucial decision in developing the HBAT v2 was to release it under the MIT license as a fully open-source project. This represents a philosophical shift from the original closed-source model and reflects evolving best practices in scientific software development. Open source ensures that the tool can continue to evolve even if I'm unable to maintain it personally, and it allows the community to contribute improvements and bug fixes.

The open-source approach also enhances reproducibility – a cornerstone of good science. Researchers can examine exactly how analyses are performed, verify the correctness of algorithms, and even modify the code if their research requires specific adaptations. This transparency builds trust and enables the kind of collaborative improvement that drives scientific progress.

Looking Forward

The scientific community's needs continue to evolve, and the HBAT v2 is designed to evolve with them. Future development will be guided by user feedback and emerging needs in the field. The modular architecture of the HBAT v2 makes it relatively straightforward to add new analysis methods, support additional file formats, or integrate with other tools in the computational biology ecosystem.

A Personal Reflection

This project has been deeply personal, representing not just a technical exercise but a commitment to the scientific community that has supported and used the HBAT over the years. Seeing how a tool created during an internship has contributed to research around the world has been gratifying. The rewrite represents an opportunity to honor that trust and ensure that the HBAT can continue serving science for years to come.

The journey from PERL to Python, from closed source to open source, and from monolithic script to modular application reflects broader trends in scientific computing. As scientific research becomes increasingly computational, the tools we create must meet higher standards of reliability, maintainability, and accessibility.

Conclusion

The HBAT v2 is now at available at GitHub with comprehensive documentation. If you are a structural biologist, a medicinal chemist, a computational researcher, or anyone working with protein-drug interactions or similar problems, I invite you to try HBAT v2. Whether you're a long-time user of the original or are discovering it for the first time, your feedback is invaluable.

Please, download it, test it, and use it in your research. Explore the API. If you find a bug, have an idea for a new feature, or have a unique use case, please open an issue on the GitHub repository.

References

[1] A. Tiwari and S. K. Panigrahi, "HBAT: A Complete Package for Analysing Strong and Weak Hydrogen Bonds in Macromolecular Crystal Structures," 2007, *SAGE Publications*. doi: 10.3233/ISI-2007-00337.

[2] A. Tiwari and V. Tiwari, "HBNG: Graph theory based visualization of hydrogen bond networks in protein structures," 2007. doi: 10.6026/97320630002028.