ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-MOBR003

VIRTUAL BEAMLINE++: ACCELERATING THE DESIGN OF HIGH-ENERGY, NEXT GENERATION LASER ARCHITECTURES

J. Penner†, R. Aden, K. McCandless, S. McLaren, L. Waxer
Lawrence Livermore National Laboratory, Livermore, CA, USA

Abstract

Virtual Beamline++ (VBL++) is an advanced nonlinear optical laser beam propagation code used for the design and operation of high-energy and high-power laser systems. It is the latest tool employed to precisely setup the National Ignition Facility's (NIF) laser beamlines and support a broad range of missions from fusion ignition to high energy density science.

VBL++ also plays a critical role in supporting a growing base of users with emerging and diverse laser designs, technologies, and applications. This tool actively contributes to the design of novel architectures, including some for Inertial Fusion Energy applications, proposed by NIF researchers and collaborators. VBL++ offers an intuitive user interface for building optical chains and supports high-resolution runs using High Performance Computing (HPC) resources. This talk will focus on a subset of features in VBL++ that help users to elevate their modeling above single simulations. These features include:

- Parameterization and optimization of inputs
- An interface for smooth desktop-to-HPC integration.
- An inverse solver used to determine the input low-power pulse shape that will achieve the high-power request on target.
- An interface written in Python, MATLAB, and IDL that enables users to build chains of optical components and run simulations with greater configurability.

This application is still under active development: new models and enhanced features are added regularly to address evolving user requirements at the cutting edge of fusion class laser design and operations.

INTRODUCTION

As laser physicists continuously seek to enhance the capabilities of their systems, it is essential to ensure that experimental configurations yield the desired outcomes without risking damage to costly equipment. To address this requirement, scientists and engineers at Lawrence Livermore National Laboratory have developed specialized software to model laser behavior and simulate energy gain as well as potential sources of damage. At Lawrence Livermore National Laboratory, precursors to the NIF facility used MALAPROP as early as the 70's [1] to validate their setups. In the mid 90's, its successor PROP92 and later PROP [2] were used in the design of the NIF facility itself. The Virtual Beamline (VBL) [3] code was developed in the mid 2000's to support the daily operations of the NIF laser, with an architecture that allowed for new optical models to be added regularly to increase the code's fidelity.

As NIF's operating range was pushed even further in the hunt for ignition, users of the VBL code required higher resolution simulations to capture all the fine-grained spatial and temporal effects in their laser setup calculations. Developers and users both concluded that a major upgrade to the code was required to allow simulations to be performed in parallel execution environments and capture more computationally intensive laser physics effects

VBL++ – named as an homage to the C++ upgrade from the C programming language – is an application written in C++ that provides an intuitive interface for users to build simulations that could represent anything from a tabletop setup to the NIF laser chain. It can then run the simulations anywhere from a personal laptop to an HPC server. This paper will explain the core functionality of the VBL++ application as well as highlight features that have been added to support the wide range of use cases that scientists require.

OVERVIEW

Modes of Propagation

VBL++ operates using a paraxial approximation of the classical wave equation [4] which allows for a split-step algorithm that separates propagation of an electric field array and nearfield interactions into two separate steps. The propagation step can be implemented by performing a Fast-Fourier Transform (FFT), applying a phase term, and performing the inverse FFT. This approximation significantly decreases the computational complexity of the propagation algorithm and minimizes interactions between data points within the grid, thereby facilitating the implementation of data-level parallelism.

This default mode of propagation – with the electric field values stored in a 3-dimensional complex array representing the electric field's shape in 2-dimensional space (with an x and y axis) and a pulse length in time – is called the "Monochromatic Fourier" mode. VBL++ also offers a further approximation named the "Hankel" mode, which assumes the laser beam has a revolution symmetry over space and therefore can be represented in a single radial dimension. The "Broadband" mode is named as such because it aims to model broadband laser effects –where a Fourier transform can be performed along the time dimension to manipulate the frequency domain– such as those captured in the unidirectional pulse propagation equation (UPPE) [5]. Because the Broadband mode often requires a highly resolved computational grid, a "Chirped" approximation mode is included that assigns instantaneous frequencies to each time slice [6].

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-MOBR003

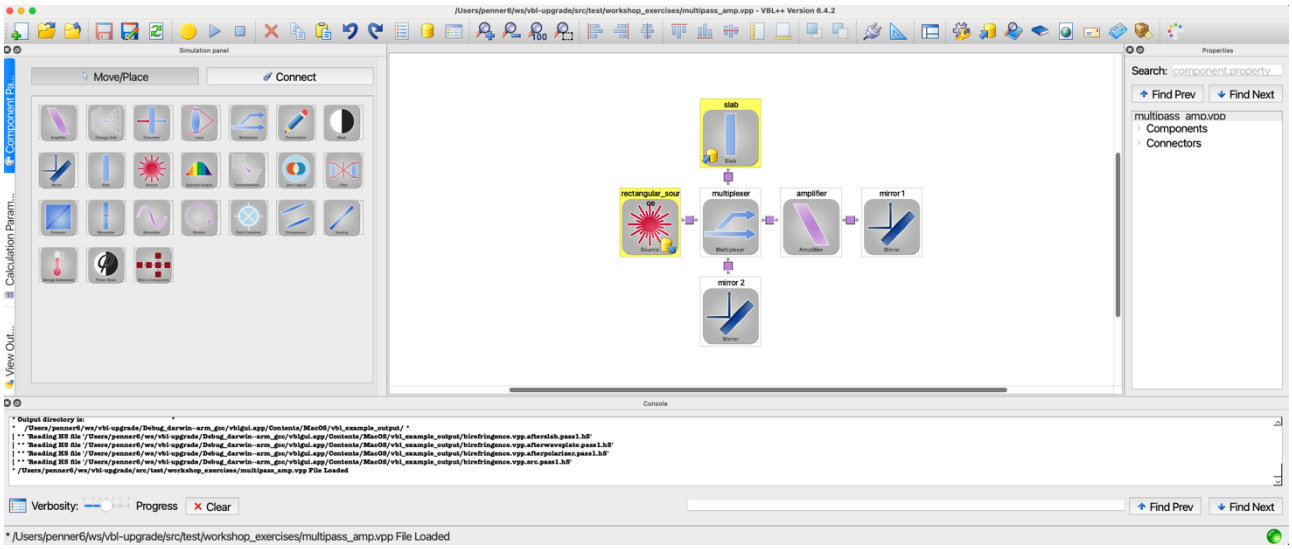

Figure 1: A depiction of a simple VBL++ deck in the GUI.

In a typical VBL++ simulation the beam propagates through a chain of optical components, which have their own models of interaction with the field. Various statistics such as the total energy over the whole field and the position and direction of the beam are captured after interacting with each component. At any point before or after interaction with a component the current state of the beam may be captured as the full electric field array, a 2 dimensional fluence profile, a 1 dimensional power profile, or several other formats and written to an HDF5 [7] file, where a user can view and interpret the data using their interactive tool of choice.

Component Models

While not an exhaustive list of optical component models that VBL++ implements, here is a list of some notable effects captured:

- Diffraction and dispersion effects while propagating through a given material.
- Other material effects such as bulk linear absorption, two-photon absorption, Kerr and Stimulated Rotational Raman Scattering.
- Beam focusing and spatial filtering, including an adaptive resampling algorithm [8] to avoid loss of information while near the focal spot and interact with objects along the focal plane.
- Spherical, cylindrical, and ellipsoidal lens shapes.
- Spherical and parabolically shaped mirrors.
- A grating component that captures pulse compression and Smooth Spectral Dispersion effects.
- Amplification through an excited gain media, modeled using a fast Frantz-Nodvik or more advanced lower-level lifetime algorithms, or a more accurate Generalized Recovery Model [5, 9].
- Second, Third, and Fourth harmonic generation through a frequency converter. Recent upgrades allow generalized sum and difference frequency generation with any user-defined material.
- An algorithm to capture potential damage over time [10].

Configuring a Deck

There are two methods to create an optical chain in VBL++. The first method is to write a file with the .vpp extension. These files use the YAML standard and provides a description of optical layout as well as any global or component specific parameters. These files are termed "decks," a reference to the historical practice of using physical punched card decks for computer simulations. The second method is to use the "drag-and-drop" graphical user interface (see Fig. 1) to build the chain with a visual aid. This interface is inspired by Miró [11], a similar laser modeling code written by the French Atomic Energy Commission (CEA). Decks written using this interface are also saved into the .vpp format, allowing seamless transition between the two modes of operation.

Every deck includes at least two sections: first, a set of global parameters that determine the grid size, mode of propagation, and regions of interest as well as overrides to turn off certain physical effects. Second, the user can build their own chain of components, always starting with a source. Each component contains information that controls either a mode of operation (for example, multiple models exist in VBL++ for the amplifier component that capture different effects during interaction) or a physical attribute such as length or angle relative to the propagation axis.

Parallelism

VBL++ was designed with the intention of modeling effects that require high spatial and temporal resolution. Approximations and adaptive step methods are used wherever available, but these mitigations cannot possibly account for the entire scope of operation. However, because each value in the electric field array is typically treated as a separate propagating "pencil" beam with minimal crosstalk, most of the models implemented in VBL++ are highly parallel-friendly.

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-MOBR003

An initial approach to improving runtime performance on both personal computers and HPC servers involves the use of thread-level parallel operations. VBL++ makes use of the RAJA open-source library [12] developed at LLNL that provides an abstract policy-based interface to create C++ executables with threading enabled using the OpenMP library. Using RAJA enables VBL++ to choose which dimensions to parallelize by defining a policy compile time. This allows the code to choose the best parallelization policy for each loop with considerations for operations that may necessitate serialization in one or more dimensions.

Sometimes VBL++ simulation resolutions go above local memory limits. To circumvent this limitation and to add an extra layer of parallelism to simulations, VBL++ can be run on distributed memory HPC systems. In this mode, the beam is divided along the Y dimension into localized chunks that run separately on each server node. In the uncommon occasion that one node requires information stored on another (e.g. the beam does a 90-degree rotation around the axis of propagation), information is passed using the Message Passing Interface (MPI).

Why Y? Generally, when distributing an array across multiple nodes, the outermost dimension is the one that is distributed to maximize the amount of contiguous memory. The electric field array is defined with the dimensions from outermost to innermost index (T, Y, X) because most operations on the field capture spatial effects, and the row-based ordering in C++ means the dimension contiguous in memory is the last one. In this case, the obvious choice for a distributed dimension would be the T axis. However, the amplifier component – which is used in most setups – involves an ordinary differential equation that captures evolution of the gain media over time. Because the evaluation of this equation requires that the loop over time must be done sequentially, the next outermost dimension was chosen to be distributed (see Fig. 2).

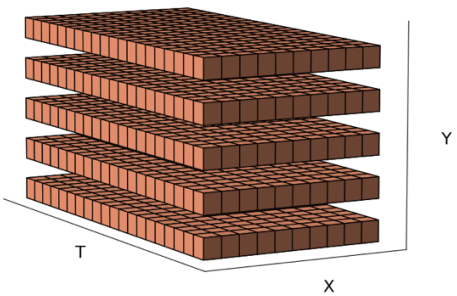

Figure 2: A visual representation of how the electric field array is split and distributed among server nodes.

NOTABLE FEATURES

Parameter Sweeps and Design Optimizations

An essential capability in optical system design, validation of experimental assumptions, and analysis of component behavior under varying conditions is the ability to execute multiple simulations with systematically varied input parameters. VBL++ offers a parameter sweep mode where a user selects one or more properties in the deck and provides a parameter space to iterate over. A variety of uniform and pseudorandom sampling methods can be used to define the parameter space.

Sweep parameters can also be used to modify multiple values at once (see Fig. 3), as well as more sophisticated approaches. For example, say a user seeks to sweep over the position of an optical component in 1 meter of space. To keep overall propagation distance the same, if the distance before the component was some factor d, then the distance after the component should be $1 - d$. VBL++ provides the equation parsing capability and the context with the sweep parameter so the user can create this exact equation.

Complimentary to parameter sweeps is the design optimization mode, that uses a nonlinear optimization algorithm with the help of the NLoPT [13] interface to determine the value of an input that maximizes or minimizes a user specified output metric. An example use case of this feature would be determining the detuning angle of a frequency converter that maximizes the energy for an output beam with a specific wavelength. The interface is nearly identical to that of the sweep parameter interface, with the addition of some parameters to adjust step size tolerances and limit the maximum number of iterations in the case that the solver fails to converge.

HPC Launcher

A typical HPC run would entail connecting to the server and running a command through the queuing service to request an allocation with the desired resources. This process necessitates familiarity with the queuing system's interface, resource constraints, and requisite command arguments. The HPC Launcher serves as an abstraction layer for users who may lack experience with the system or the Linux environment yet wish to utilize the parallel computing capabilities of the cluster.

From their personal computer, a user can bring up an interface to open an SSH connection with a remote server. Once the connection is established, the user can select from a few parameters such as the number of nodes to use and launch the job. The simulation inputs are copied over to the server and run with the VBL++ instance on the server. A table is provided that shows all jobs and their status (pending, running, completed, or failed). Once the job is completed, the user clicks a button that copies the output files back to the local filesystem.

This interface currently assumes that jobs on the remote server are scheduled using SLURM. Plans have been made to expand the capabilities of this interface and make it more robust against potential failures such as connections that were terminated early.

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-MOBR003

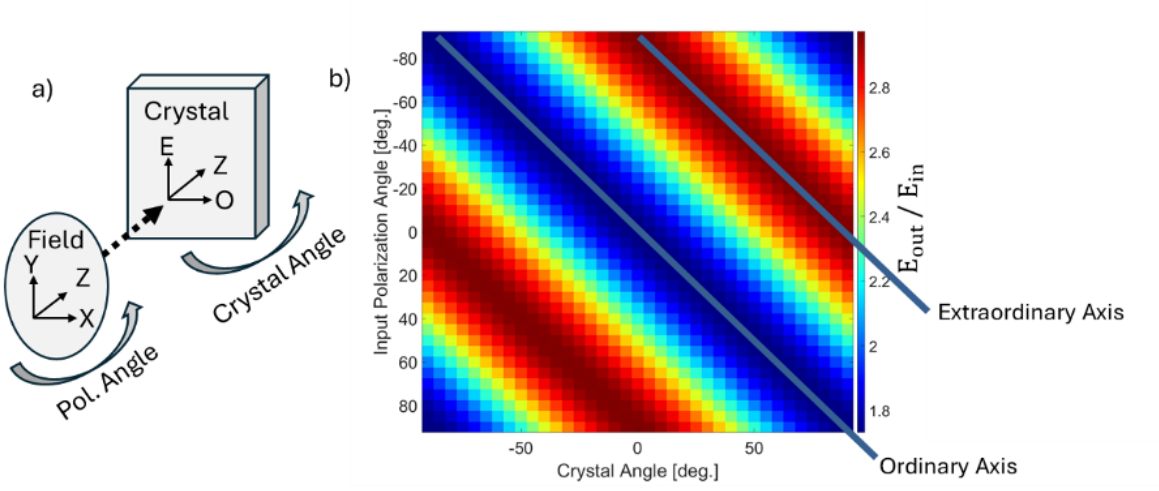

Figure 3: An example use case of the parameter sweep feature. a) describes the case: the user wanted to test the robustness of the VBL++ crystalline gain material for any possible input polarization angle and amplifier crystal state. b) shows the results of the parameter sweep simulation.

Pulse Solver

In the design of experiments performed using the NIF laser, physicists need to know how to shape the temporal shape of the laser pulse to achieve the pulse shape at the output of the NIF optical chain. Due to the presence of nonlinear effects in certain components of the optical chain, a direct inverse simulation is not feasible. The pulse solver was developed as a method to obtain an input pulse shape given an output pulse shape without the necessity of inverting these effects.

The solver begins with developing an "initial guess" by either accepting an input pulse shape from the user or running a series of Nt low-resolution, simplified simulations that builds the shape of the guess. The algorithm then starts iteratively running a simulation of the current pulse shape through the optical chain, comparing the output pulse shape to the desired shape to see if it is correct within tolerance, and then adjusting the input shape for the next iteration using an inverse Frantz-Nodvik calculation. This iteration continues until a solution is converged upon or the maximum number of allowed iterations is exceeded.

This method of determining an input pulse shape has proved crucial for laser physicists to design experiments in the context of NIF. As the user base of VBL++ expands over time, the use cases for the pulse solver have expanded as well. These use cases challenge assumptions made by the pulse solver. For example, the solver assumed that at least one amplifier would exist in the optical chain, and therefore the gain of the beam would be greater than 1. Adjustments to the pulse solver are currently being implemented to capture the new use cases.

Script API (ScrAPI)

Users frequently seek to model optical chains containing components that are not yet represented within the existing VBL++ models. Additionally, there is often a desire to automate sequences of VBL++ simulations or to integrate simulation and analysis into a single streamlined process. For this purpose, an interface to VBL++ was created in the Python, MATLAB, and IDL languages. ScrAPI has emerged as one of the most widely utilized additions to the software package in recent years.

The API is written separately in the three languages to remove interdependencies, but the interface is maintained to be as similar as possible. The implementation creates a layer of abstraction over a YAML parsing library to allow users to add, remove, or modify values in a .vpp deck file. It also has functions to set paths and run simulations once the file has been written.

An example use case for ScrAPI stems from an assumption made by VBL++ for simplicity which assumes the propagation direction is always on the positive or negative z axis in the global frame. Because of this choice in convention, a periscope cannot be properly modeled with just the application alone. ScrAPI, however, can be used to run a subsection of a periscope setup until the mirror is reached that would reorient the propagation axis, take the output and model the necessary flips in orientation or phase effects created by the mirror interaction, then insert the beam as a file into a separate VBL++ chain that models the next section of the periscope.

FUTURE WORK

A primary objective of the VBL++ project is to fully leverage the capabilities of HPC resources. Recent interest has focused on enabling execution on GPU-accelerated hardware, particularly following preliminary tests that demonstrated substantial runtime improvements when executing the propagation algorithm on a local GPU compared to CPU-based thread acceleration. RAJA provides policies to run loops on GPU hardware that utilizes the same loop interface as for OpenMP threading. However, there are conventions to GPU device code that were not originally accounted for when the loops were first written. Affected

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-MOBR003

areas of code are currently being refactored with device code guidelines in mind.

New physics models are regularly proposed to expand the range of operation. Here are some that have recently been brought to focus:

- A complex model that captures effects when the front of a laser pulse overlaps with its tail end in an amplifier.
- A new model for propagation beyond the standard paraxial approximation.
- Expanded support for polarization effects, including better integration with COMBINE [14], a physics code that generates matrices capturing material dependent polarization effects.
- Modeling volumetric temperature effects in materials.
- Exploration into the interactions of different effects – for example, adding the Generalized Recovery Model to the Broadband amplifier algorithm.
- Updates to allow for a true 3D propagation convention so that configurations such as the previously mentioned ring cavity can be properly modeled.

CONCLUSION

VBL++ has proven itself a key tool in supporting laser physicists in the era of laser-induced ignition. It is used daily in the calibration of the NIF laser, and it is used in the design of improvements and even successors to NIF. The code was designed with extendibility and flexibility in mind, allowing for new physics models and supporting tools to be continuously developed and added to the workflow. Because of this design, VBL++ is well equipped to support laser physicists in the development of future successors to the state-of-the-art laser facilities.

This software has become an indispensable resource for physicists at NIF, as well as for collaborators at Lawrence Berkeley National Laboratory and the University of Rochester's Laboratory for Laser Energetics. These recent collaborative efforts have helped the team to understand assumptions made in the context of NIF that don't necessarily hold in other laser chain setups. They have also brought new ideas for updates to existing models and providing new models for optical chain interactions. The project aims to broaden its user base and further enhance the mutual influence between VBL++ and the wider laser physics community.

ACKNOWLEDGEMENTS

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Release Number: LLNL- CONF-2011007.

REFERENCES

[1] W.E. Warren, "MALAPROP user's guide", Lawrence Livermore Laboratory, Livermore, CA, USA, Aug. 1977. https://doi.org/10.2172/5208558

[2] R. A. Sacks, M. A. Henesian, S. W. Haney, and J. B. Trenholme, "The prop92 fourier beam propagation code", ICF Quarterly Report 6(4), Lawrence Livermore National Laboratory, Livermore, CA, USA, Report UCRL-LR-105821-96-4, pp. 207-213, 1996.

[3] R. A. Sacks *et al.*, "The virtual beamline (VBL) laser simulation code," in *Proc. High Power Lasers for Fusion Research III*, A. A. S. Awwal and M. A. Lane, Eds., San Francisco, CA, USA, Feb. 2015, p. 93450M. doi:10.1117/12.2084848

[4] J. D. Jackson, *Classical Electrodynamics*, 3rd ed., Hoboken, NJ, USA: Wiley, 1999.

[5] M. Kolesik and J. V. Moloney, "Modeling and simulation techniques in extreme nonlinear optics of gaseous and condensed media," *Rep. Prog. Phys.*, vol. 77, no. 1, p. 016401, Dec. 2013. doi:10.1088/0034-4885/77/1/016401

[6] S. A. McLaren *et al.*, "Virtual Beamline++: full-chain optical modeling for modern laser design and operation," in *High Power Lasers for Fusion Research VIII*, C. L. Häfner and A. A. Awwal, Eds., San Francisco, CA, USA, Mar. 2025, p. 8. doi:10.1117/12.3046817

[7] M. Folk, G. Heber, Q. Koziol, E. Pourmal, and D. Robinson, "An overview of the HDF5 technology suite and its applications," in *Proc. EDBT/ICDT 2011 Workshop on Array Databases (AD '11),* Association for Computing Machinery, New York, NY, USA, Mar. 2011, pp. 36–47. doi:10.1145/1966895.1966900

[8] E. Feigenbaum, R.A. Sacks, K.P. McCandless, and B.J. MacGowan, "Algorithm for Fourier propagation through the near-focal region," *Appl. Opt.*, vol. 52, pp. 5030-5035, 2013.

[9] F.T. Yee, "Effect of finite lower level lifetime on Q-switched lasers," *IEEE J. Quantum Electron.*, vol. 24, no. 12, pp. 2345–2349, 1988. doi:10.1109/3.14358

[10] C. W. Carr, J. B. Trenholme, and M. L. Spaeth, "Effect of temporal pulse shape on optical damage," *Appl. Phys. Lett.*, vol. 90, no. 4, Jan. 2007. doi:10.1063/1.2431705

[11] O. Morice, "Miro´: Complete modeling and software for pulse amplification and propagation in high-power laser systems," *Opt. Eng.*, vol. 42, no. 6, p. 1530, Jun. 2003. doi:10.1117/1.1574326

[12] D. A. Beckingsale *et al.*, "RAJA: Portable Performance for Large-Scale Scientific Applications," in *Proc. 2019 IEEE/ACM Int. Workshop on Performance, Portability and Productivity in HPC (P3HPC)*, Denver, CO, USA, Nov. 2019, pp. 71–81. doi:10.1109/p3hpc49587.2019.00012

[13] H. Gu, J. Yi, Z. Lian, J. Tao, and X. Yan, "NLoPT: N-gram Enhanced Low-Rank Task Adaptive Pre-training for Efficient Language Model Adaption," in *Proc. 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, Torino, Italia, May 2024, pp. 12259–12270.

[14] M. Rehak and J.M. Di Nicola, "COMBINE*: An integrated opto-mechanical tool for laser performance modelling," in *High Power Lasers for Fusion Research III*, A. A. S. Awwal and M. A. Lane, Eds., San Francisco, CA, USA, Feb. 2015, p. 93450K. doi:10.1117/12.2080403