Tidy3D: hardware-accelerated electromagnetic solver for fast simulations at scale

June 27, 2022

In this white paper, we will introduce "Tidy3D", Flexcompute's simulation tool for electromagnetic modeling. Implementing one of the most general and widely used algorithms for electromagnetics (FDTD), it is applicable to an extremely wide range of systems and devices. The software is designed for maximum user-friendliness by combining the flexibility and simplicity of a scripting-based interface with the performance of advanced computing hardware in the cloud. All aspects of the implementation, from the underlying algorithm to the adaptive allocation of compute resources, are tailored and optimized for the highest performance on Flexcompute's hardware. The resulting product finds solutions 100x faster and can handle 100x larger problems than conventional CPU-based solvers. Through Tidy3D, engineers and researchers can iterate through designs orders of magnitude faster and model problems previously out of reach, resulting in a dramatic acceleration to research and product development.

1 Introduction

1.1 Finite-Difference Time-Domain

Tidy3D is an electromagnetic solver based on the "finite-difference time-domain" (FDTD) method [13]. FDTD directly solves Maxwell's equations, which are the fundamental description of the physics of classical electrodynamics [7]. This generality allows FDTD to model a huge range of phenomena, making it the most popular method for electromagnetic simulations.

In FDTD, one first discretizes space and time into a staggered rectangular grid, called a Yee Lattice [16]. Then, the electric and magnetic fields, $\vec{E}(\vec{r},t)$ and $\vec{H}(\vec{r},t)$, are solved time step by time step and ultimately used to determine other quantities of interest in the system, such as power flux or modal amplitudes. Because it is a time-domain method, FDTD gives the full dynamics of the system with no approximations required beyond those introduced by discretization. As such, FDTD simulations give a direct look into what is happening inside the system, which can be very useful for research and design purposes. FDTD is also often used to measuring the steady-state behavior of systems as it is often a faster and more reliable alternative to frequency-domain methods, such as FEM and FDFD. Enhancements to FDTD for modeling more complex phenomena, such as frequency-dependent (dispersive) material properties, nonlinear materials, and other specialized systems, are a widely studied area and many of these enhancements are included in Tidy3D's implementation.

1.2 Implementation

Tidy3D provides an interface to running high performance FDTD simulations on the cloud through a python-based API. This API provides the following core functions:

- Defining the global simulation characteristics and parameters, such as size, boundary conditions, and other options.
- Specifying the objects and materials that make up a device and interact directly with the electromagnetic waves.

- Adding sources, which inject electromagnetic energy into the system.
- Defining monitors, which instruct the solver on the kind of results and data to measure and return to the user.

The specifications needed to fully define an FDTD simulation are defined through the API and written to a single file, which is sent via the same python API to Flexcompute's servers and trigger a solver run. Once a simulation specification file is received, the optimal hardware configuration is determined to optimize performance for the given specifications and the simulation is then run. The simulation results, as determined by the monitors, are packaged into a single data file, which is sent back to the user through the same python API for further processing.

From the user perspective, this entire process is seamless and can be done from within a single python session. The definition, management, and post processing of the simulation results feel as if everything is done locally, even though it is utilizing advanced computing hardware on the cloud. For added convenience, a web-based interface provides additional tools for monitoring and visualizing a run, viewing past simulations, and managing projects.

1.3 Overview

This white paper will cover the following contents to further explain the details behind how Tidy3D works and how one can utilize the performance it provides for design or research work. In Section 2, we first give an overview of the main features included in Tidy3D, which enable modeling of a wide range of electromagnetic systems. Next, in Section 3, we outline the core functions and design principles of the user interface. After this, in Section 4, the performance of Tidy3D for both fast and large scale simulations is discussed, along with some details about how it is accomplished. Finally, in Section 5, we provide a handful of examples of Tidy3D being used in real world applications before concluding in Section 6.

2 Features

This section gives an overview of the main features present in Tidy3D for electromagnetic modeling. The features are broken up into "solver" features, which expand the range of simulations handled by the solver and "usability" features, which make Tidy3D easier to use in practice.

2.1 Solver Features

First, we will discuss some of the core features that enable the modeling of various phenomena that are present in electromagnetic systems.

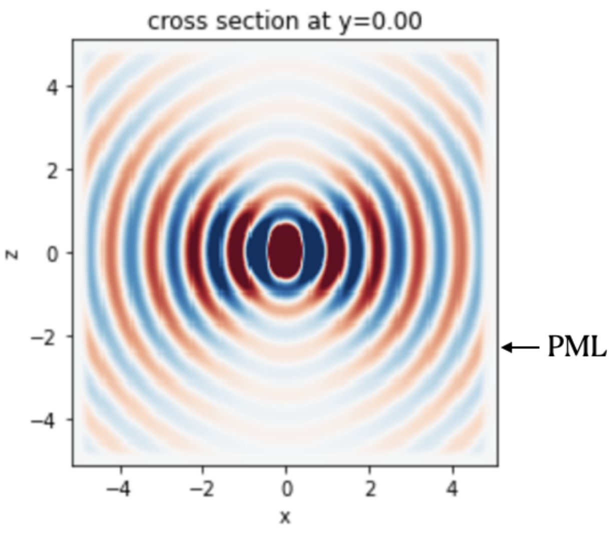
2.1.1 Absorbing Boundaries

Any FDTD code must handle the solution of the electromagnetic fields at the boundaries of the simulation domain. A default choice for many FDTD solvers, including Tidy3D, is to use a simple periodic boundary condition in which the field patterns on one edge are continuous with respect to the opposite edge. In practice, this choice of boundary condition emulates a structure that is repeating infinitely along that axis.

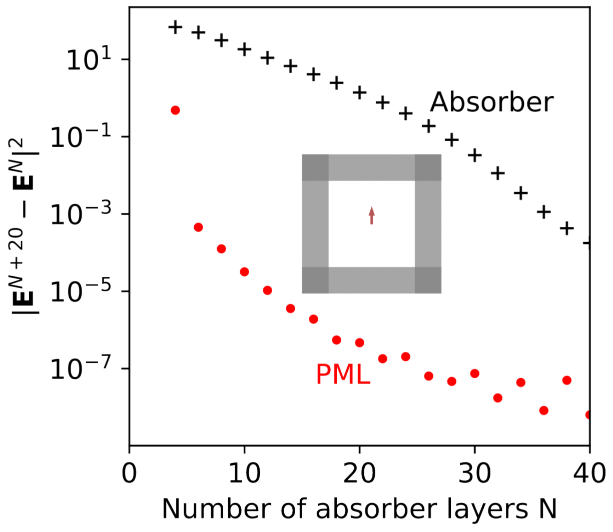
However, for many cases, one instead would like to simulate an *isolated* structure or device, which can be accomplished using “absorbing boundary conditions”. The typical approach to implementing an absorbing boundary condition in FDTD is to append a few layers of lossy material onto the edges of the simulation domain. These layers are intended to absorb any incoming waves before they can reach the true simulation boundaries.

However, there are technical difficulties with this approach. The impedance mismatch between the background medium and the absorbing layers will generally cause the layers to reflect the incoming power rather than absorb it. A standard solution is to implement a gradual increase of the amount of loss as the distance from the simulation boundary is increased. This is often referred to as an "adiabatic absorber" as the intention is to increase the loss parameter slowly enough that reflections due to an abrupt change in material properties are avoided [11].

Although this approach works, it requires a very large amount of extra cells to be added to the computational domain along each dimension. These extra cells can greatly increase the total number of unknowns of the simulation



(a)



(b)

Figure 1: (a) Example of a field radiating from a dipole source with PML boundary conditions on all sides. (b) The difference in the electric field **E** from a point source inside a 3D cube between simulation with $N+20$ absorber layers and N absorber layers, showing that the PML absorbs incoming waves with far fewer layers needed than the absorber.

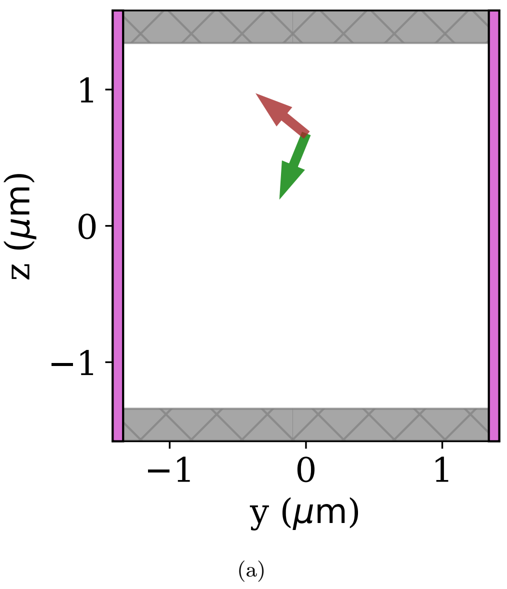
relative to without any absorber, which can negative impact performance. Thus, there has been much effort in developing absorbing boundary conditions that can be just as effective with much fewer layers. One such category of absorbers are referred to as "perfectly-matched layers" (PML). While there exist several classes of PML, each with their own pros and cons, Tidy3D implements a "convolutional" PML, as described in Refs. [12, 2]. This choice of PML has the advantages of being (a) highly absorptive of evanescent waves, (b) applicable to various host media including lossy, inhomogeneous, and dispersive media, and (c) computationally efficient. As shown in Fig. 1 (a), the field radiated by a dipole source is terminated at the boundary with no visible reflection. Compared to an adiabatic absorber, Fig. 1 (b) shows that much fewer absorbing layers are needed in the PML boundary than in the adiabatic absorber boundary.

2.1.2 Bloch Periodic Boundaries

While regular periodic and absorbing boundaries are sufficient for most common use cases, they are still not able to model periodic structures, such as gratings, in the presence of a plane wave with angled incidence. For this, a "Bloch" boundary condition is required, in which the field solution is continuous across the opposite edges of the simulation with an added phase to account for the angled incidence. While a typical FDTD solver uses real-valued fields, the Bloch formalism requires complex-valued fields to properly model the phase discontinuity.

Fig. 2 demonstrates the use of Bloch boundaries in Tidy3D to inject a plane wave at an angle of 30° with respect to the vertical; the setup is shown in Fig. 2a. Bloch boundaries are defined along the x and y directions, while PMLs are specified along the z direction. The real part of the x component of the resulting frequency-domain electric field is plotted in Fig. 2b at the frequency for which the Bloch boundary condition was defined. The angle of incidence is correctly modeled, and there are no visible fields generated “behind” the plane of incidence.

As an additional note, Tidy3D allows the independent specification of the boundary conditions on each of the six edges of the simulation domain, which can be useful for specific modeling scenarios. These boundary conditions can include perfectly conducting boundaries, in which the tangential electric or magnetic fields are set to zero to model the presence of conductors, such as ground planes or cavity walls. As an example, Fig. 3 shows the z component of the electric field for a Gaussian beam incident on, and reflected off, a perfect electric conductor



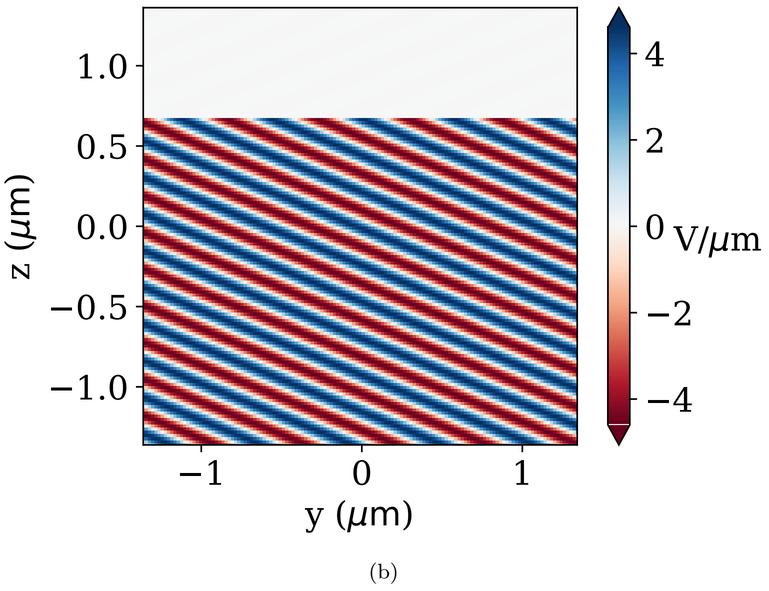


Figure 2: Demonstration of angled plane wave incidence with Bloch boundaries. (a) The plane wave is incident at an angle of 30° with the vertical. (b) The real part of the x component of the frequency-domain electric field show the expected phase distribution along the horizontal axis.

boundary on the right-hand side. As expected, since this component is tangential to the perfect electric conductor boundary, its value approaches 0 at the boundary.

For more details on setting up boundary conditions in Tidy3D simulations, see this example from the documentation.

2.1.3 Symmetry

A simulation can exhibit one or more symmetries if all structures **and** all sources are invariant under a symmetry operation. Such symmetries can be exploited to significantly cut down the number of unknowns by avoiding redundant calculations. In Tidy3D, the user can define symmetry with respect to reflection in the planes bisecting the simulation along each of the x, y and z axes separately. In the case where all three symmetries are used, the simulation has 8× fewer unknowns and will therefore generally run 8× faster and with 8× less memory compared to the original case. When one or more symmetry values are specified, the user-facing operation of Tidy3D is identical, yet the solver will make use of this information to cut down on time and memory. This is illustrated in Fig. 4 for a simulation of a plane wave impinging on a periodic array of dielectric boxes. Note that the structure is symmetric with respect to all three planes. However, the source breaks the symmetry in the z-direction, so only symmetry along x and y can be applied. Furthermore, when specifying a symmetry plane, it is required to provide the eigenvalue of the reflection of the **E**-fields, which can be either $+1$ ("PMC" symmetry, tangential **H** fields vanish on the plane), or -1 ("PEC" symmetry, tangential **E** fields vanish on the plane). This eigenvalue is determined by the source, in this case the polarization of the plane wave, but needs to be supplied by the user. As can be seen by the brown arrow in Fig. 4(b), which denotes the E-field component of the source, there is a positive symmetry with respect to the $x = 0$ axis, and a negative symmetry with respect to the $y = 0$ axis. The fields obtained with and without applying these symmetries are identical as shown in Fig. 4(c)-(d).

For more details on exploiting symmetry in Tidy3D simulations, see this example from the documentation.

2.1.4 Nonuniform Mesh

In finite-difference discretization schemes, such as FDTD, the space is divided along the Cartesian axes (x,y, & z). The spatial step sizes along these axes (Δx, Δy, Δz) can be chosen independently, in case a different resolution is

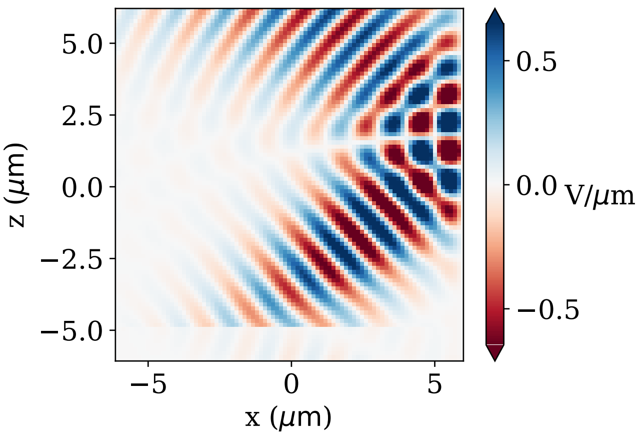


Figure 3: A Gaussian beam reflecting off a perfect electric conductor boundary on the right-hand side.

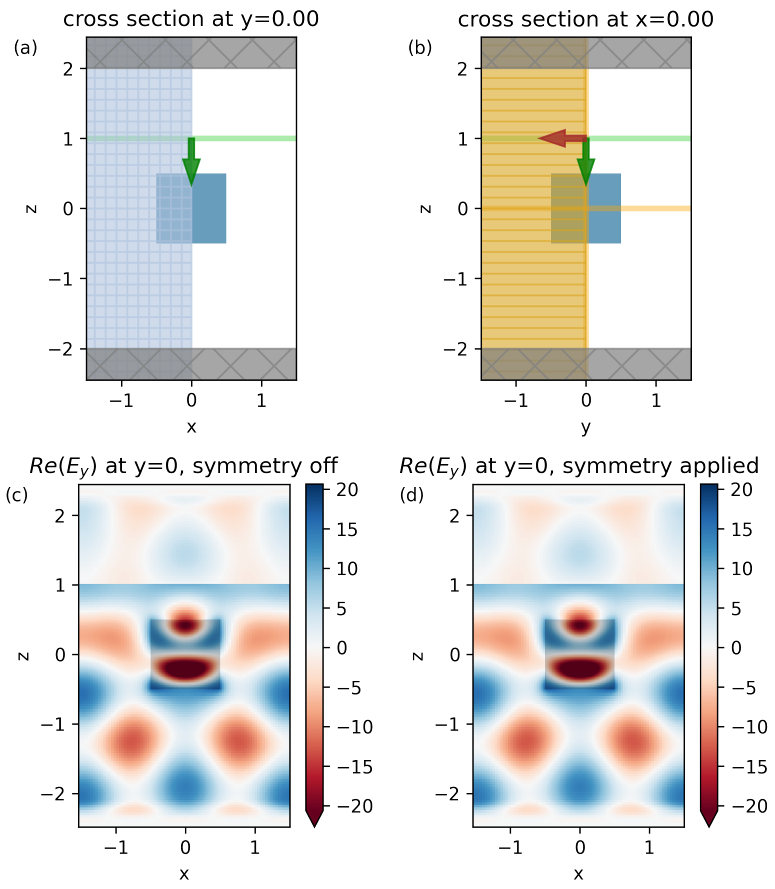


Figure 4: (a)-(b) Cross sections at $x = 0$ and $y = 0$, respectively, for a simulation of a plane wave impinging on a periodic array of dielectric boxes. Symmetries can be applied to the simulation at both the $x = 0$ and $y = 0$ planes. The symmetry eigenvalue for those planes is determined by the polarization of the source. (c)-(d) Simulated electric field without and with explicitly applying symmetry in the simulation.

required along one or more of the axes.

However, more fine tuned control of the spatial resolution can be immensely beneficial for some applications where there may be specific regions of the simulation that require a finer grid resolution. For example, regions containing smaller features or higher index materials will generally require more grid cells than those comprised of empty space. To provide this freedom, Tidy3D includes a "nonuniform mesh" feature in which the grid sizes may vary along each dimension, eg. $\Delta x = \Delta x(x)$, $\Delta y = \Delta y(y)$, $\Delta z = \Delta z(z)$. As shown in Fig 5 (a) The use of a nonuniform mesh leads to a reduction in the number of grid cells for a given accuracy, which can dramatically speed up the simulation time and memory requirements.

The proper determination of the grid sizes $\Delta x(x)$, $\Delta y(y)$, $\Delta z(z)$ requires many considerations and depends highly on the specific simulation geometry. For example, it is important to be able to choose a nonuniform mesh that satisfies global resolution requirements, such as the minimum number of grid cells within a wavelength or skin depth within a material. With many structures distributed within a domain, finding the optimal nonuniform grid specifications to satisfy these constraints while minimizing the total number of grid cells can be a challenging optimization problem. Additionally, it is often much more accurate for simulation boundaries to coincide with the natural boundaries of structures within the simulation. A good choice of nonuniform mesh will therefore often seek to make the grid commensurate with any such edges along a given dimension. Third, it is important that the size of adjacent grid cells does not change too abruptly, otherwise this can cause instabilities in the solver.

To make the process of determining and deploying a nonuniform mesh as easy as possible, Tidy3d provides an automatic meshing tool that can satisfy constraints specified by the user for a given simulation, an example is shown in Fig. 5 (b). The automatic nonuniform meshing can be straightforwardly applied to any simulation, with the user simply providing general resolution requirements without needing to worry about the details involving the discretization. For more control, the ability to provide "override structures" allow users to specify regions of space to mesh more or less finely without these structures affecting the actual simulation space. Alternatively, a custom nonuniform mesh may be supplied if a specific one is desired by the user.

2.1.5 Sub-Pixel Averaging

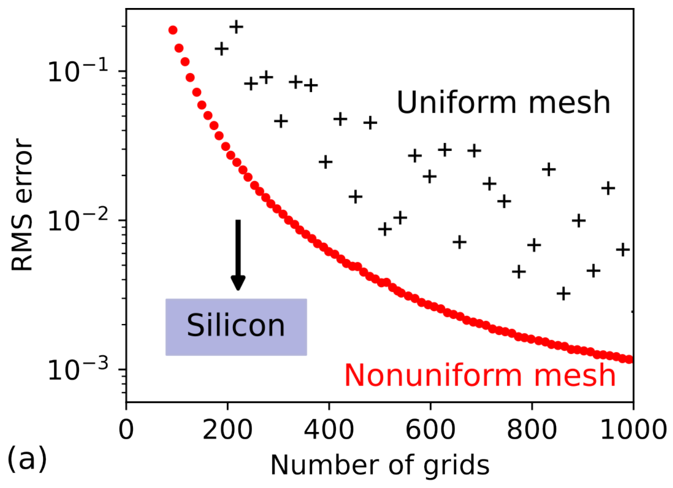
The standard discretization scheme used in FDTD places each of the x, y, and z components of the electric and magnetic fields at different positions of the rectangular volume associated with each grid cell. As the material properties may vary within a single cell, to accurately model complex geometries, each field component within a cell will generally experience a different value of the permittivity and permeability tensors $\epsilon(r)_{i,j}$ and $\mu(r)_{i,j}$ evaluated at that position.

However, one can further increase the accuracy of a simulation by augmenting this material assignment to make greater use of the characteristics of the structure geometry, including the filling fraction and normal direction within each cell. For example, if a structure boundary bisects a cell with a normal oriented in the x direction, it may be useful to know what percentage of that structure lies within the cell and assign an appropriate material property that is an intermediate value between the structure material and its surrounding material. Neglecting such information can lead to surprising results as adjusting the continuously varying geometric parameters will lead to discontinuous and artificial "snapping" to the spatial grid.

To solve this issue, Tidy3D includes an automatic "sub-pixel" smoothing algorithm [9] when rendering structure geometric and material properties on the simulation grid. An basic illustration of the scheme is shown in Fig. 6(a). The main benefit of such a scheme is that the FDTD results are far more accurate for a given resolution. It follows that simulations with sub-pixel smoothing turned on can be run at a lower resolution, making them faster and requiring of less memory. In Fig. 6(b), we give an example of the improvement that sub-pixel smoothing can have in terms of simulation performance and accuracy.

2.1.6 Mode Solver

FDTD is used widely in integrated photonics applications, where light is guided through a device using dielectric waveguides and other structures defined on a planar substrate. For these applications, it is often useful to describe the wave propagation in terms of the "modes" of the waveguides and other components. The mode fields describe stable solutions to Maxwell's equations assuming translational invariance in some direction. A typical simulation setup will have a number of input-output channels, as well as components that mix and couple those channels, or "ports". Many devices are most conveniently understood through the scattering matrix defining how light



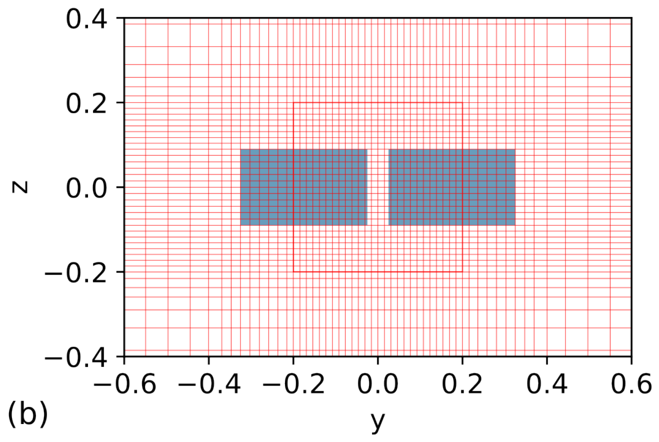
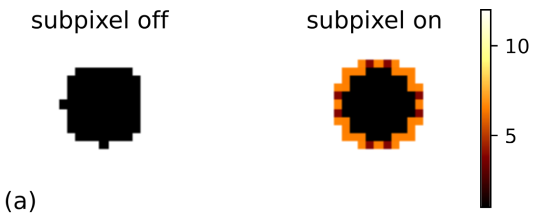


Figure 5: (a) Accuracy of the FDTD solver when modeling transmission through a silicon slab vs. the number of grid cells when using uniform mesh (black plus) vs nonuniform mesh (red dot). Nonuniform mesh is shown to be consistently more accurate for a given number of grid cells. (b) The cross section of a waveguide coupler showing the automatic nonuniform mesh (red lines) with an override structure (red square) added to force higher resolution within the central coupling region.



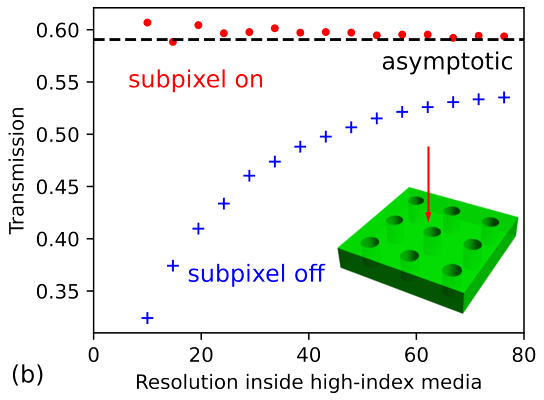


Figure 6: (a) Illustration of the relative permittivity along the out-of-plane direction of a cross section of a simple photonic crystal slab geometry [inset in (b)] evaluated on the simulation grid with and without sub-pixel smoothing. (b) Results showing accuracy vs. grid resolution with and without sub-pixel averaging.

incoming from each port is transmitted to all the different modes of all the other ports. This abstracts away the complex physics of large devices and allows for component-level modeling, as we will discuss further in a later section.

In Tidy3D, the solution of the electromagnetic modes of a waveguide is done using a frequency-domain mode solver that is completely separate from FDTD. For this purpose, Tidy3D includes a "mode solver" plugin that is integrated directly with the rest of the solver for ease of use. The mode solver uses the same spatial discretization as the time-domain solver to ensure the lowest possible numerical mismatch when the two are used together. A simulation object and a 2D cross-section may be supplied to the mode solver, along with additional specifications that tell the solver the modes to look for. Examples of supported features include modes in straight, angled, or curved waveguides (Fig. 7). The mode solver may be run either locally or through a web interface to return the field patterns and propagation characteristics of the modes in the system. From this information, a mode source can be added to a Tidy3D simulation which would inject that specific mode into the simulation (green components in Fig. 7). Similarly, mode decomposition monitors can be added (orange components in Fig. 7). If such a mode monitor is supplied, the frequency-domain fields on the monitor plane are computed during the simulation run. After that, these fields are decomposed in forward and backward propagating amplitudes associated with each mode using the inner product ([14], chapter 4):

$$(\mathbf{u}_1, \mathbf{u}_2) = \frac{1}{4} \int_A (\mathbf{E}_1^* \times \mathbf{H}_2 + \mathbf{E}_2 \times \mathbf{H}_1^*) \cdot \mathrm{d}\mathbf{A}, \tag{1}$$

where $\mathbf{u} = (\mathbf{E}, \mathbf{H})$ combines both electromagnetic fields, the integration is over the monitor plane area A, and $\mathrm{d}\mathbf{A}$ is the surface normal. In both mode sources and monitors, all mode fields are normalized such that $(\mathbf{u}_m, \mathbf{u}_m) = 1$. Then, if we denote the fields recorded in the mode monitor during the simulation run by $\mathbf{u}_s$, the complex power amplitude carried by mode m is given by the coefficient $c_m = (\mathbf{u}_m, \mathbf{u}_s)$, while the power carried by that mode is given by $|c_m|^2$.

For more details on the use of the Tidy3D mode solver, see this example from the documentation.

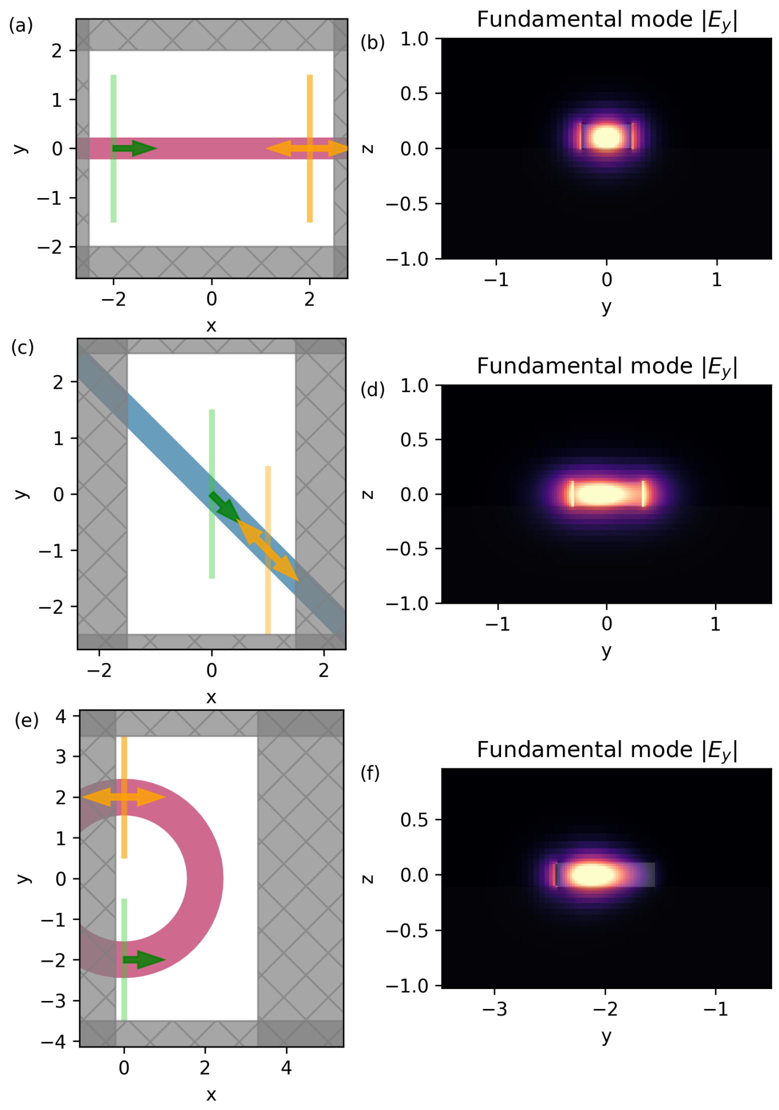


Figure 7: (a) Simulation setup involving a straight dielectric waveguide and (b) its corresponding mode profile. (c)-(d) Angled waveguide and mode profile. (e)-(f) Curved waveguide and mode profile.

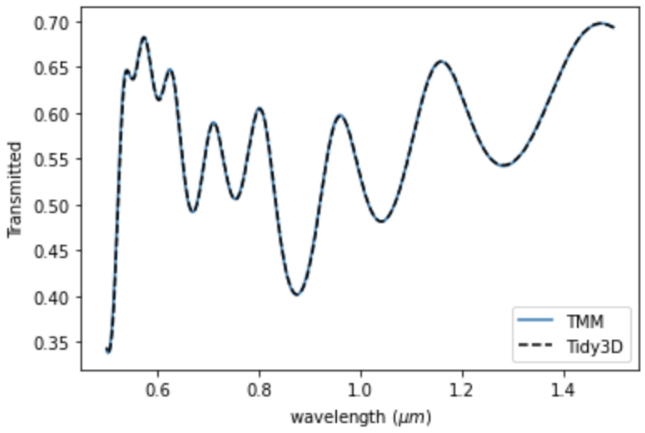


Figure 8: Transmission spectrum through a slab of dispersive materials calculated with transfer matrix method (blue) and Tidy3D (dashed black), showing very good agreement.

2.1.7 Dispersive Materials

In general, the optical properties of materials vary as a function of frequency, a phenomenon known as "dispersion". While it is convenient to neglect this property, it is often required for the accurate modeling of electromagnetic devices. In contrast to frequency-domain methods, where the material properties at each frequency can be directly evaluated and fed to the solver one by one, modeling of dispersion in the time-domain can be challenging as it requires careful accounting of auxiliary field quantities [3] and also lead to divergence of the simulation. In fact, the interaction of the PML with some structured dispersive medium where opposite phase and group velocities are supported is well known to cause divergence issues in FDTD [8], and can be a major barrier to modeling of many integrated photonic devices with dispersion.

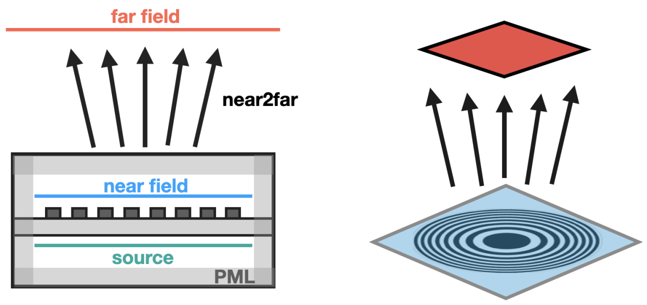
Tidy3D has developed many improvements to both the handling of dispersive materials and absorbing boundary properties to be able to run these situations accurately and without divergence. As for the dispersive materials, we are employing the complex-conjugate pole-residue pairs to more efficiently model dispersive medium [3]. As for the absorbing boundary, our convolutional PML implementation can provide effective absorption at the boundary in the presence of dispersive medium. In Fig. 8, We give an example of a simulation with our handling of dispersive materials compared to that with transfer matrix method. As shown, the careful handling of dispersion and absorption layers in Tidy3D can enable modeling of dispersion for highest accuracy in many applications. In situations where PML is for sure to fail and even to result in divergence [8], our implementation of adiabatic absorber boundary can be used to improve the stability of simulations.

For more details on dispersive materials in Tidy3D, see this example from the documentation.

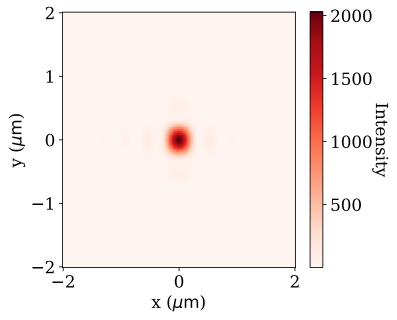
2.1.8 Near Field to Far Field Transformation

In many applications, especially those involving free-space optics, one may want to solve for the electromagnetic fields far away from a device without including that additional space explicitly in the simulation. For example, in antenna design, one may wish to examine the cross sectional scattering properties of the device in the far field limit while a modeling of only the relevant features of the antenna. For this, it is often convenient to combine FDTD simulations with a far field transformation tool. Tidy3D provides one such method, called a "near field to far field transformation". This tool takes measurements of the electromagnetic fields on a plane or series of planes near the structure and uses them to project the fields to an arbitrary set of points in the far field, which were not contained in the original simulation. This approach can dramatically cut down on the computational cost of the simulation because it avoids needing to model the empty space between the device and the far field.

The near field to far field transformation tool first constructs a set of equivalent current densities associated with the electric and magnetic fields near the device. If injected to a homogeneous system without the device, these current densities would generate an equivalent outward propagating field pattern, and are therefore useful for projecting to the far field. Applying a Fourier transform to these near field current densities leads to the calculation



(a)



(b)

Figure 9: (a) Basic setup of a near field to far field transformation: the near fields are recorded on a small simulation and then the fields at a set of points outside of the simulation can be computed via projection. (b) Results of a near field to far field transformation for a zone plate simulation, showing focusing at the focal plane.

of the "propagation vectors", which from each near field surface to the projection points and observation angles. From these propagation vectors we can then compute the scattering cross section or electromagnetic fields in the far field limit.

In Fig. 9, we describe the basic setup of the near field to far field transformation and give a simple example: a zone plate designed with its focal plane at a distance of $32\,\lambda_0$ along the axis of the device, where λ_0 is the wavelength in free space. As indicated in Fig. 9a, the near fields are recorded just above the device, and then the associated surface current densities are used to generate far fields on the patch shown in red, $32\,\lambda_0$ away from the device. The resulting field intensity on the focal plane in the far field is shown in Fig. 9b, where the focusing effect of the zone plate is apparent.

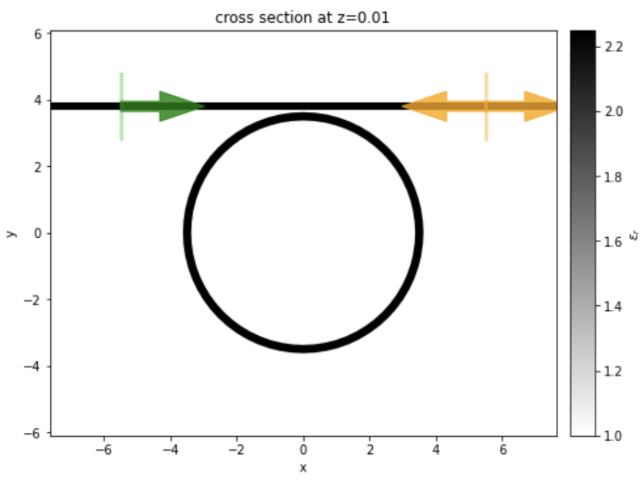
2.2 Usability Features

In addition to the solver features described in the previous section, Tidy3D provides numerous additional tools that allow users to seamlessly define and analyze their simulations.

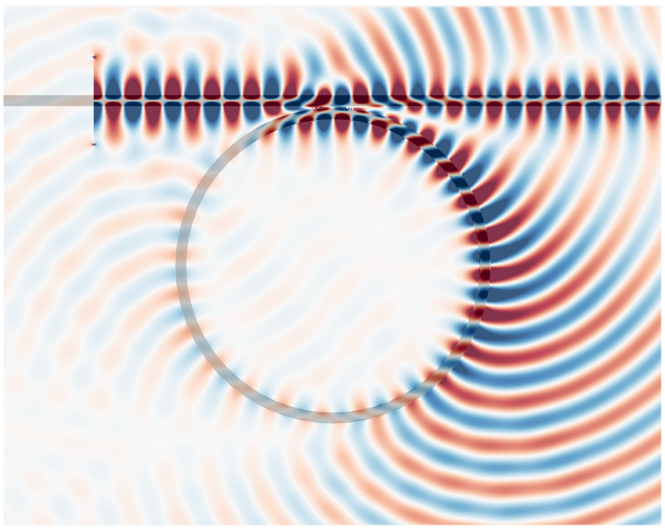
2.2.1 Visualization

Detailed visualization of the simulation is crucial to verifying a correct setup before submitting for solution. Tidy3D provides numerous ways to visualize both individual components and entire simulations through the python API. Additionally, more complex visualizations are provided through the web interface, which can be viewed interactively before running the simulation.

The data returned by an FDTD simulation can be very complex, so being able to visualize it is important to verify that the correct results were obtained As Tidy3D uses monitors to specify what data gets returned, this data is neatly organized by monitor. Each monitor has a well-defined data type, which can be simply visualized both in the web interface and as a data set loaded in the python script. An interactive app may also be used to



(a)



(b)

Figure 10: (a) The geometry of a ring resonator simulation being visualized through the python API before submission. (b) The resulting field pattern being visualized the same python API after loading the solver data.

scan and select various dimensions of the data, making it easy and intuitive to inspect the data both locally and on the web before beginning a more detailed post-processing stage. Examples of both pre and post run visualization are shown in Fig. 10

For more details on simulation and data visualization in Tidy3D simulations, the following documentation examples may be of interest: simulation plotting, data plotting.

2.2.2 Dispersive Material Parameter Fitting

As described previously, the optical properties of materials generally vary as a function of frequency and the modeling of such dispersion properties can be challenging in time-domain methods such as FDTD. Often times, dispersion is modeled in terms of coefficients corresponding to a material model, which may be determined by a least error fit to experimental data. Tidy3D provides a simple and convenient tool for fitting such material models to optical measurements and the resulting models may be converted to Tidy3D components for use in the simulation. To ensure the stability of FDTD simulation, we have implemented our proprietary stability criterion in our fitting tool. Consistently stable FDTD simulations are observed when materials are fit using this method.

In the dispersion fitting plugin, the user can supply raw data describing the frequency-dependence of the material in the form of a text file, url link, or through direct specification in a python script. Tidy3D then fits this data to our native dispersive models using a nonlinear least squares optimization. General fitting parameters, such as the maximum complexity of the resulting model or constraints on the stability requirements of the model, can be easily specified. The result of the fit is a component that can be directly fed to a Tidy3D simulation to be used as a material. Fig. 11 shows the result of such a dispersive fit on user-supplied data.

For more details on setting up the dispersion fitter plugin in Tidy3D simulations, see this example from the documentation.

2.2.3 Material Library

In many cases, one does not need to fit raw optical data and would instead rather load a dispersive model for their given material. To make this as seamless as possible, Tidy3D provides dispersive models for several dozen commonly used metallic and dielectric materials that can be directly loaded and used in simulations.

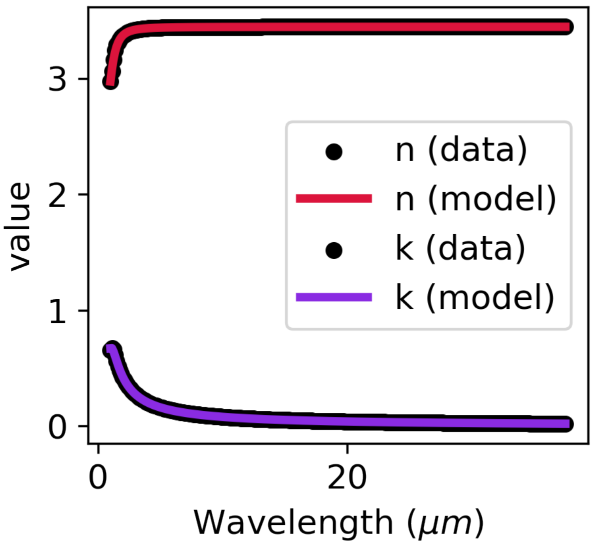


Figure 11: Results of the dispersive material fitter on fitting user-supplied optical data for the real and imaginary (n, k) parts of the refractive index. Dots represent the raw data and the solid line represents the model fit after x iterations of the nonlinear solver. The resulting material can be loaded directly into a Tidy3D simulation.

This material library is populated using peer reviewed data and provides several options to choose from for many of the materials, depending on the frequency range of interest.

For a complete list of supported materials and their variants, refer to the material library guide on our documentation.

2.2.4 Component Modeler

For systems-level modeling of photonic devices, often times it can be useful to have a higher-level description of each of the components to be able to construct a "combined" model involving the connection of several such component models. A typical approach is to form a "scattering matrix" for each component, which relates how a series of "ports" are connected in a linear system. For example, the i-th row of the scattering matrix of a device would represent the outgoing amplitude of each of the ports if an amplitude of 1 were injected into the i-th port. This is an especially natural way of modeling integrated photonic devices, where the ports can represent the modes of various waveguides on the surrounding edges of the device.

Tidy3D includes a tool for solving the scattering matrix of a device through a "component modeler". One first defines a "base" simulation describing the device, along with the "ports". With this information, the tool will automatically construct and run one simulation per port, using the results to construct the scattering matrix. An illustration is shown in Fig. 12 for a waveguide coupler and the resulting scattering matrix is displayed in Fig. 13.

For more details on setting up boundary conditions in Tidy3D simulations, see this example from the documentation.

2.2.5 Web Interface

While Tidy3D has a python-based interface, it also includes a rich companion web interface that provides several functions, such as

- managing Tidy3D account and credits.
- storing past simulation specifications, status and results.

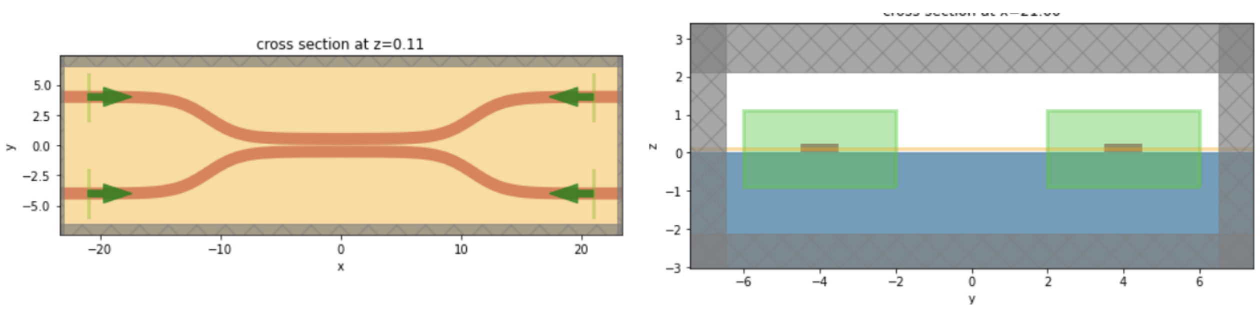


Figure 12: Illustration of the component modeling tool being used to find the scattering matrix of a four port splitter. (left) Four simulations are constructed out of the "base" simulation, representing a modal source injected into each of the ports, represented by the green arrows. (right) Cross-sectional view of the ports situated on the waveguides.

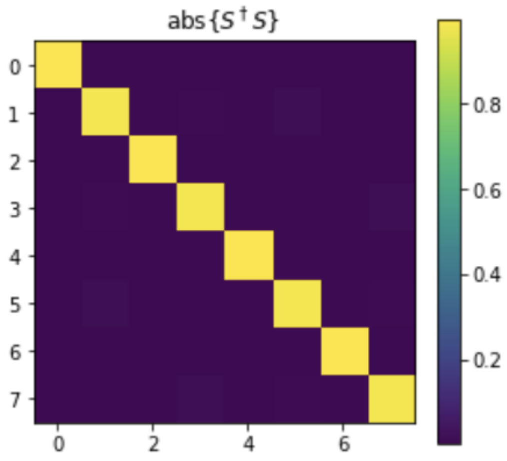


Figure 13: Scattering matrix S obtained through the mode amplitude measurements from each of the the 4 port simulations illustrated in Fig. 12. As expected for a lossless, reciprocal system, the result of $S^\dagger S$ (as plotted here) is close to the identity matrix.

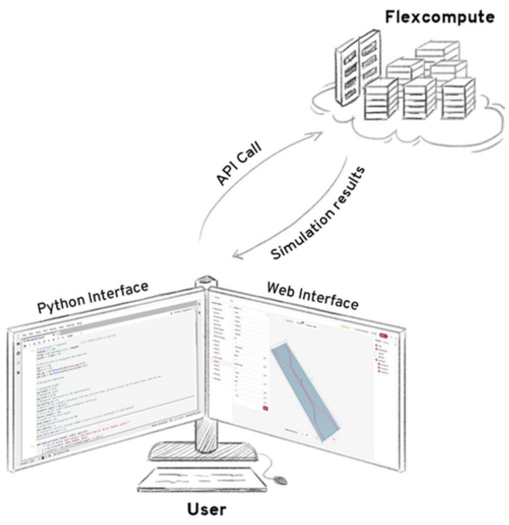


Figure 14: How the front end fits with the rest of the solver.

- visualizing 3D simulation structures.
- visualizing simulations results post run.

When users sign up for an account, access to this web interface is automatically included.

3 Interface

In this section, we will give a summary of the user-facing interface to Tidy3D and its design principles. Tidy3D simulations are defined and submitted using a python package, the documentation of which can be found here. This package provides tools for:

- Defining the full specification of a Tidy3D simulation.
- Submitting simulations to Flexcompute servers, monitoring their progress, an loading the results.
- Post-processing and performing analysis of the resulting data.
- Numerous plugins for generation of components and processing of data.

This process of the python front end interacting with Flexcompute servers is visualized in Fig. 14.

3.1 Defining the Simulation

The first step of any Tidy3D simulation is to fully define the specifications of the simulation. Tidy3D makes use of the python package "pydantic" to define a simulations as a single data structure that can be programmatically generated with precise specifications. Some of the advantages of this approach are:

- Rigorously defined fields, no ambiguity about accepted values.
- Schema can be generated directly from the simulation model and its components.
- Validation of the simulation contents are preferred upon creation, making it very difficult to incorrectly specify a simulation without triggering an error.
- Straightforward saving and loading of simulation specifications from file.

In the following subsections, we will describe the various components that go into defining a single Tidy3D simulation model.

3.1.1 General Parameters

First, the user must specify the general parameters describing the simulation context. The most critical pieces of information are:

- The size and center of the simulation domain.
- Specifications describing the discretization.
- Boundary conditions.
- Absorbing boundary layers or PMLs.
- Presence of symmetry.
- Maximum simulation run time.
- Presence of sub-pixel averaging.
- The material properties of the background space (if not vacuum).

3.1.2 Structures

The general parameters outlined in the previous section describe the basic "canvas" of an FDTD simulation, but describing the actual device to be modeled is done by defining "structures". A structure in tidy3d combines one or more "geometry" specifications with a "medium" specification. The geometry simply describes how the structure is laid out in space whereas the medium describes the optical properties of the material that the structure is made of. One structure may have several geometries assigned to the same medium through the use of a "geometry group", which can speed up pre-processing operations.

A device in a tidy3d simulation is made of a collection of structures specified as a list. The material assigned to any region with overlapping structure will default to that of the later structure in the list, allowing one to create complex geometries through overlay of structures. Planar geometries, such as photonic integrated circuit layouts, may also be defined through the import of a GDS file for added convenience.

3.1.3 Sources

The next critical components of every simulation are "sources". Sources define the electric and magnetic current present in the simulation, which inject electromagnetic energy into the system. One can specify the source directly, for example by defining an electric dipole at a point or a sheet of current on a plane. Alternatively, one may define the source in terms of the desired field pattern, for example a plane wave, Gaussian beam, or waveguide mode. The python package provides an interface for defining several types of sources, which can be independently created and added to the simulation as a list, similar to structures.

3.1.4 Monitors

While the FDTD simulation will solve for the full $\vec{E}(\vec{r},t)$ and $\vec{H}(\vec{r},t)$ at each time step, one generally only needs a small subset of this data from the solver for analysis or visualization purposes. Therefore, the final core component of Tidy3D simulations are "monitors", which allow users to specify how the solver should measure and package the data. The various types of monitors can be broken into

- "field monitors", which measure the electric or magnetic fields directly.
- "flux monitors", which measure the electromagnetic flux through a plane.
- "mode monitors", which measure the amplitudes corresponding to the modal decomposition of the electromagnetic fields on a plane.
- "permittivity monitors", which measure the permittivity distribution generated by the sub-pixel smoothing algorithm.

Both field and flux monitors can be specified in the time or frequency domain, whereas mode monitors may only measure data in the frequency domain. Frequency-domain measurements are done using a running discrete Fourier transform, meaning that the frequency components are computed on the fly rather than after post processing a time series, leading to significant memory savings. The data returned by the Tidy3D solver is separated by monitor, which can be accessed using the monitor "name" parameter, specified as a string.

3.1.5 Simulation

Each of the previous components are combined into a single simulation object, which fully characterizes a Tidy3D simulation that the solver can process. An example of a very simple example of a simulation specification is:

```python
import tidy3d as td

sim = td.Simulation(
    size=(4, 4, 4),
    run_time=1e-12,
    grid_spec=td.GridSpec.auto(
        min_steps_per_wvl=20
    ),
    structures=[td.Structure(
        geometry=td.Sphere(
            center=(1, 0, 0),
            radius=0.5,
        ),
        medium=td.Medium(permittivity=2)
    )],
    sources=[td.PointDipole(
        center=(-1, 0, 0),
        source_time=td.GaussianPulse(
            freq0=200e12,
            fwidth=20e12
        ),
        polarization='Ex',
    )],
    monitors=[td.FluxMonitor(
        center=(0, 0, 0),
        size=(td.inf, td.inf, 0),
        freqs=[200e12],
        name='flux'
    )]
)
```

Listing 1: simulation definition

3.2 Managing Simulation Runs

3.2.1 Web API

Once the simulation is defined, it must be submitted to Flexcompute servers to run the solver. A series of tools are provided in the python API that give the user the ability to interface with our job scheduler for the purposes of

- Uploading a simulation.
- Inspecting metadata about an uploaded simulation.
- Triggering the solver to run a simulation.
- Monitoring the progress of a simulation.
- Downloading results and loading them into the python session.

These tasks are accomplished by the "web API", which is a series of functions imported from the "web" sub-directory of Tidy3D.

3.2.2 Batch Processing

In many cases, one would rather submit a collection of runs rather than manage them individually. For this application, the web API also provides batch processing functionality, which can accept a dictionary mapping of task name to simulation specification and run these tasks concurrently.

3.3 Post-Processing of Data

The output of a Tidy3D simulation is a dataset that combines the individual data associated with each of the supplied monitors. This dataset can be downloaded as a file in ".hdf5" format and loaded in the python interface as a "simulation data" object. Square bracket indexing with a monitor name is used to select data for a specific monitor, for example:

```
field_data = sim_data['field']
```

Listing 2: accessing monitor data

Tidy3D makes use of the "xarray" [4] package to assign coordinates to the individual monitor data objects. For example, a field monitor may contain a dataset containing a 4 dimensional array of complex-valued floating point numbers where the dimensions correspond to x-position, y-position, z-position, and frequency of each if the field measurements. Using xarray, we may assign coordinates to each of these four dimensions, giving the user the ability to select and interpolate the data at arbitrary slices within the data. In the previous example, the user may want to select the field value at the origin for frequency "f", which can be done quite simply as

```
field_data.Ex.sel(f=f).interp(x=0, y=0, z=0)
```

Listing 3: selecting and interpolating data

Data associated with a batch of simulations can be similarly loaded as a "batch" data object or iterated through similar to a python dictionary, which is more appropriate for situations where it is not feasible to load the data from all simulations into local memory at once. For example:

```
for task_name, sim_data in batch_data.items():
    ex_data = sim_data['field'].Ex.sel(f=0)
    ex_center = ex_data.interp(x=0, y=0, z=0)
    print(f'simulation {task_name}:'
    print(f'has Ex of {ex_center} at origin.')
```

Listing 4: loading batch data

4 Performance

In this section, we discuss the factors that influence the performance of Tidy3D in terms of speed and scale.

4.1 Speed

The main competitive advantage of Tidy3D is that it performs simulations orders of magnitude faster than other FDTD solvers. While this may seem like magic on the surface, there are actually some fundamental reasons behind this accomplishment.

4.1.1 Hardware and Software Co-Design

It's well understood that there is a trade off between generalization and performance. A machine that is designed for a specific task will usually outperform one that is designed to handle many different tasks. The same is true for computing. The CPU in one's laptop is the ultimate generalist, able to handle many complex instructions, but this ability is wasted on repetitive and predictable problems, such as the many mathematical operations found in simulations. Many of the major developments in computing hardware have been in the form of new processing units that are specialized to perform certain repetitive tasks extremely efficiently. For example, GPUs were developed for ray tracing computation in graphics, but were found to perform well in many other mathematical computing tasks in linear algebra.

In Tidy3D, this concept is taken one step further. Since we are tasked with solving Maxwell's equations alone, it is possible to design and optimize all of the computation around this specific task. Thus, by sacrificing even more generality, Tidy3D accomplishes world class speed and performance.

4.1.2 Harnessing Parallelism

Most of the interesting problems in electromagnetics require large simulations on the order of millions to billions of grid cells. To perform these simulations in a reasonable time frame, one inevitably needs to exploit massive parallelization. By having many specialized processors working together, Tidy3D can typically scale all simulations down to the minute scale, regardless of their size. The result is a seamless experience where the user can receive the results of even an enormous simulation within the same session with a fast enough turnaround time to get design work done promptly.

However, with massive parallelization come massive technical challenges that must be overcome. For example, each processor needs information from the others to proceed with the computation. As one adds more processors, the amount of information needed to transfer between processors likewise increases until it becomes the limiting bottleneck in speed. As such, Tidy3D has many optimizations in place that are designed to carefully handle this aspect of the computation.

Leveraging Flexcompute's expertise in high performance computing, Tidy3D's algorithms and hardware strategies work together to push back this fundamental communication bandwidth limit far beyond where it would be for a naively-written solver. Doing so allows one to add more and more processors to large problems and scale down the simulation time significantly.

In many physics simulations, there is a notion of "locality" in which two spatial regions of the simulation can be considered completely separately within the amount of time required for information to travel between them. It is possible to exploit this fact to design algorithms that divide work among processors in both spatial and time dimensions to minimize the frequency and total amount of information transfer. Algorithms such as SWEPT [1] can overcome this communication bottleneck and offer a significant performance improvement.

4.1.3 Adaptive Computation

As no two simulations are created equal, much effort in Tidy3D has been spent on analyzing the performance of various simulations to determine the optimal hardware configuration for a given task. Since Tidy3D's solver technology can be efficiently deployed in the cloud, Flexcompute's servers are able to seamlessly allocate hardware adaptively to give the best possible experience with optimal performance. The user needs to know nothing about the computation to make the most out of the computation platform, as everything is handled adaptively by Flexcompute's servers.

4.2 Scale

Full-wave electromagnetic simulations are the most accurate way to model complex device physics and ultimately arrive at good designs. In addition to being able to solve electromagnetic problems much faster, Tidy3D is also able to simulate very *large* devices. This ability enables a new simulation paradigm in which complicated devices with many components may be simulated in one shot, instead of needing to break them up into sub-components to be modeled separately. With this capability, one can accurately model entire device workings without fear of throwing away valuable information regarding the system as a whole.

Tidy3D users report being able to run simulations that are so large that they would be not only extremely slow, but even *impossible* to run on conventional FDTD solvers. They are able to study the system at a level of detail that was previously out of reach, which often leads to the uncovering of novel results and insight.

4.2.1 Parallelization

To solve simulations at such scale requires an efficient implementation of massive parallelization for the problem at hand. Every simulation domain must be decomposed into sufficiently small problems that can be solved within the compute and memory limitations available to each processor. As simulations scale to enormous sizes, the bottleneck to massively parallel computation becomes the communication between computation domains. Traditional CPU-based parallel computing, due to its architecture, is practically unable to handle the amount of massive parallelization needed for such devices as it is unable to overcome the communication speed bottleneck.

Tidy3D's solver overcomes the speed bottleneck for massive parallelization by tightly integrating the advanced computation hardware with the software design. This coherently co-designed system is specifically optimized for solving Maxwell's equations. By exploiting knowledge about the physics describing the electromagnetic system, Tidy3D intelligently decomposes the large problem in ways that minimizes the communications between computation domains.

The result in solver achieves both weak and strong linear scaling to the simulation size. As such, Tidy3D can adaptively allocate however many resources are needed to handle a problem of a given size with a communication bottleneck limit that is far beyond what most conventional solvers can achieve.

4.2.2 Adaptive Resource Allocation

This ability is made accessible to the average user by hosting the Tidy3D solver technology on the cloud, providing 1 TB of memory access in a high speed connected network. This removes the barrier for users to access the significant hardware resources necessary to solve such enormous problems. Additionally, it alleviates the cost of purchasing and maintaining such a large amount of hardware through adaptive allocation of hardware resources tailored to each simulation. By providing a cloud-based solution, the user has access to all of this technology and engineering without ever needing to think about it.

The final result of all of this work is an electromagnetics solver that can seamlessly handle problems with billions of grid points within minutes, which is unachievable with traditional massively parallel CPU-based computation hardware. As we'll describe in the following section, this capability fundamentally opens up new opportunities for modeling devices at a scale and accuracy never possible before.

5 Case Studies

Finally, in this section we will present a handful of examples of Tidy3D being used in real world projects.

5.1 Waveguide Bend Design

A recent validation for a customer working in integrated photonics involved modeling transmission through a waveguide bend with a radius of several 10s of microns. This problem exhibited an enormous physical size compared to the optical wavelength, translating into 2 Billion of grid points, and 64,000 time steps. Traditional FDTD is practically unproductive to handle problem of this size, even with massively parallel CPU-based computation

The customer was running the simulations on a 96-core instance running conventional FDTD, with each simulation requiring 27 hours. Using Tidy3D, they were able to obtain the same results in just over 3 minutes.

5.2 Anderson Localization of Light

Collaborators at Yale University recently used Tidy3D to detect the existence of Anderson localization of light in a numerical simulation, which has significant ramifications for the electromagnetic theory community [15]. The study involved the simulation of light transmission through a slab consisting of a random distribution of spheres.

Because of the statistical nature of the study, thousands of such simulations needed to be run to extract the necessary proof of localization. Simulations of this size would typically take days on typical computing platforms, but were completed in minutes using Tidy3D, enabling a study of this scale for the first time.

5.3 Large Area Metalens Simulation

In a recent publication, we simulated a large-scale three-dimensional metalens of $70\mu m$ diameter including the focal length [5]. This seemingly tiny device is a enormous physical size for full-wave simulations compared to the optical wavelength, which is about 100x100 λ^2 in plane, and 46 λ along the optical axis. At $\lambda/18$ resolution, this translates into approximately 2.7 Billion grid cells and 6,235 time steps.

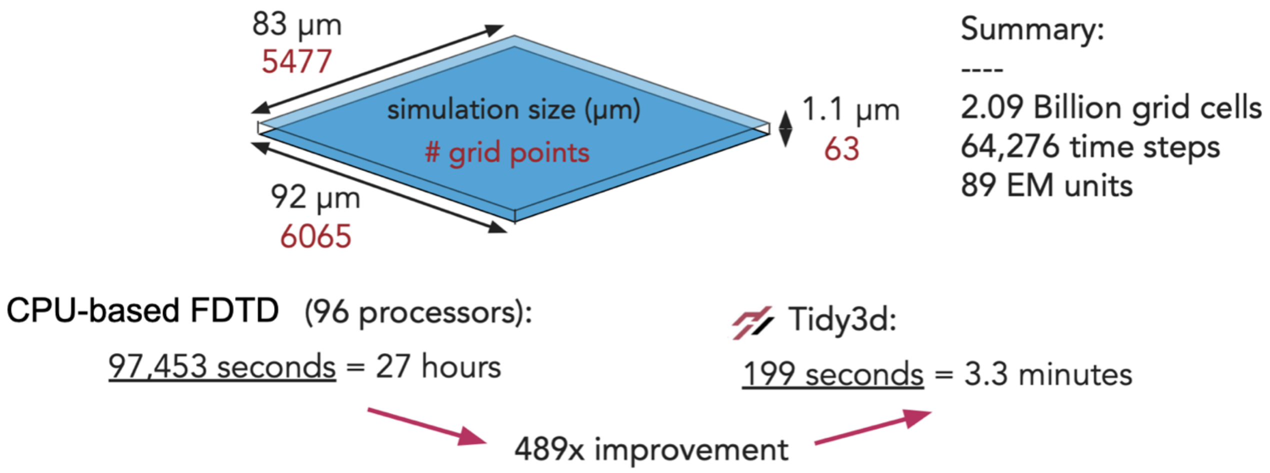


Figure 15: Comparison between CPU-based FDTD and Tidy3D on waveguide bend geometry.

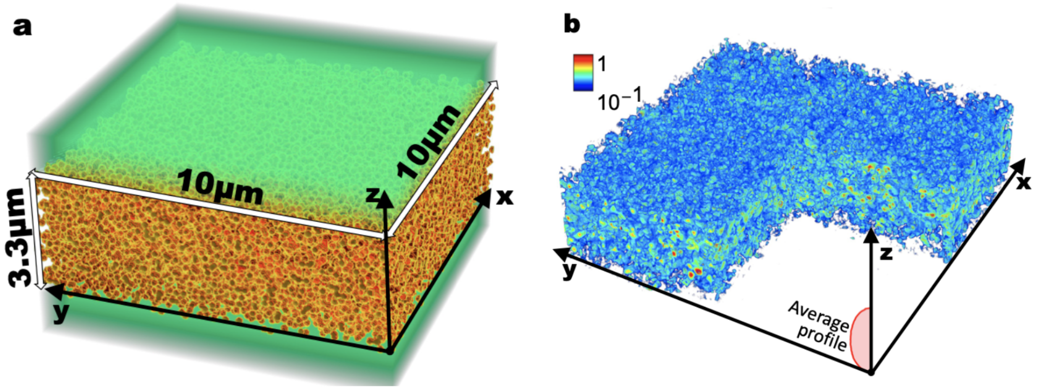


Figure 16: Anderson localization study. (a) A simulation of light propagation through a slab containing thousands of randomly distributed spheres. (b) Field intensity data measured via Tidy3d and used to find signal of Anderson localization of light in 3D for the first time in a numerical simulation. Each simulation took 45 minutes with Tidy3d, instead of 40 hours on 100 core CPU, enabling large scale parameter scan.

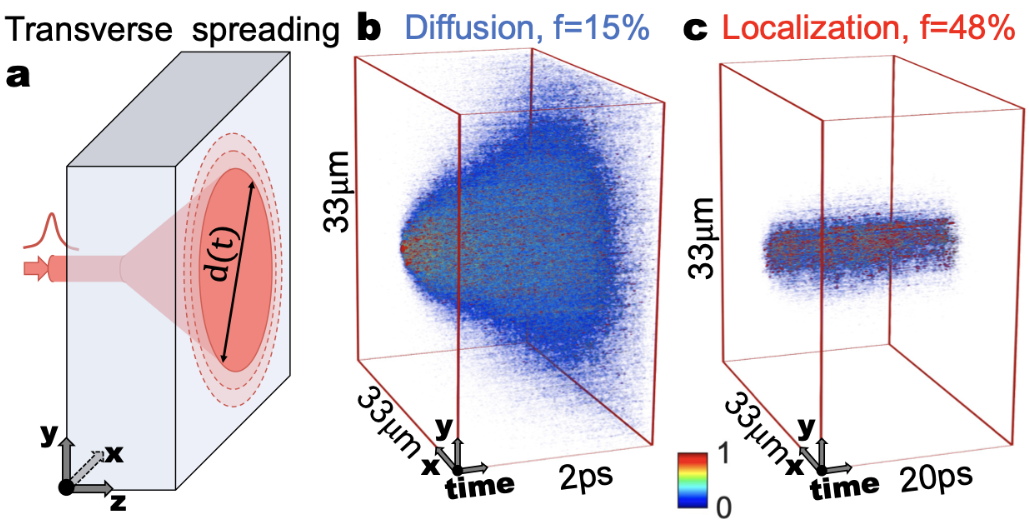


Figure 17: One piece of evidence found in support of Anderson localization in 3D electromagnetic waves. (a) Schematic of typical beam spreading when transmitted through slab. (b) Beam spreading in diffusive (non-localized) parameter regime. (c) Beam remaining columnated in Anderson localized parameter regime. These regimes were determined by running thousands of simulations to pinpoint the localization regimes.

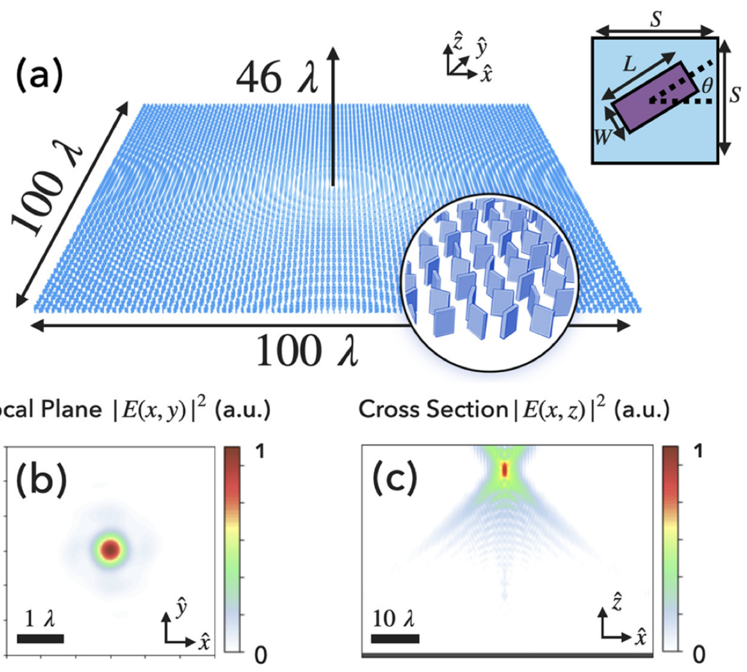
A problem of this size would be practically impossible to handle using conventional FDTD, even with massively parallel CPU-based computation. With Tidy3D, this record-sized simulation finishes in less than 5 minutes and we can readily scale the size to dozens of billions of grid cells.

5.4 Adjoint Optimization

In recent years, the concept of "inverse design" has become very popular as a method of optimizing photonic devices through simulation [10]. In inverse design, one defines a device figure of merit (FoM) to optimize with respect to a number of design parameters in the system. A mathematical technique known as the "adjoint method" is used to compute the gradient of the FoM with respect to each of the design parameters by combining a "forward" simulation of the system with one additional "adjoint" simulation encoding information about the figure of merit [6]. With this gradient information, one can perform iterative, gradient-based optimization, leading to a locally optimal set of design parameters in only a handful of steps, no matter the number of design parameters. This approach enables novel photonic devices to be constructed with highly unintuitive designs based on pixelated grids or complex curved geometries. In contrast, a brute force optimization approach quickly becomes infeasible as the number of free parameters grows, limiting the design space to conventional devices.

Here we demonstrate an example of inverse design using Tidy3D to optimize a three-dimensional integrated photonic device operating as a mode converter. The device has an input (left) and output (right) waveguide supporting both a fundamental and higher order mode. We inject the fundamental mode into the input waveguide and wish to design the central region to convert the maximum amount of power into the higher order mode of the output waveguide. The central design region consists of an 8×12 array of dielectric blocks with permittivity varying from vacuum to the waveguide permittivity. As shown in Fig. 21b, we use the original and adjoint simulations to calculate the gradient of the mode conversion FoM with respect to the permittivity of each pixel at a given iteration. Then, we perturb the material properties of each of the pixels using this gradient information, and repeat for 30 steps until convergence. As shown in Fig. 21, the final device displays an unintuitive geometry but converts almost all of the power successfully to the higher order mode.

Even though this was a three-dimensional device requiring 60 total simulations, because of Tidy3D's speed, the final result was obtained in under 1 hour. For more details on using adjoint-based optimization to compute gradient and optimize photonic devices, please refer to the example in the Tidy3D documentation.



(a)
46 λ
100 λ
100 λ
S
L
θ
W
Focal Plane $|E(x,y)|^2$ (a.u.)
(b)
1 λ
Cross Section $|E(x,z)|^2$ (a.u.)
(c)
10 λ
1
0

Figure 18: (a) A record-size metalens simulated in under 5 minutes with full-wave accuracy using Tidy3d. This particular simulation required 2.68 billion spatial grid cells. (b) The intensity pattern measured at the focal plane and (c) the side cross section of the device.

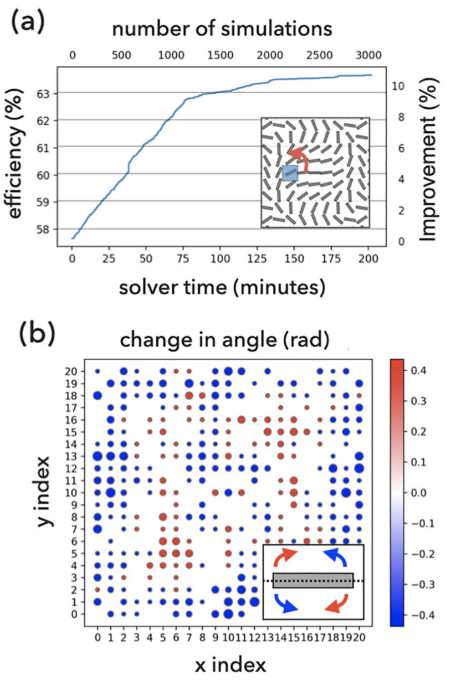


Figure 19: Results from a brute force optimization of a smaller metalens, including (a) the improvement of the focusing efficiency of 10% in under 2 hours and 1500 simulations. (b) The optimal perturbations applied to a standard starting device design that lead to such an improvement in device performance.

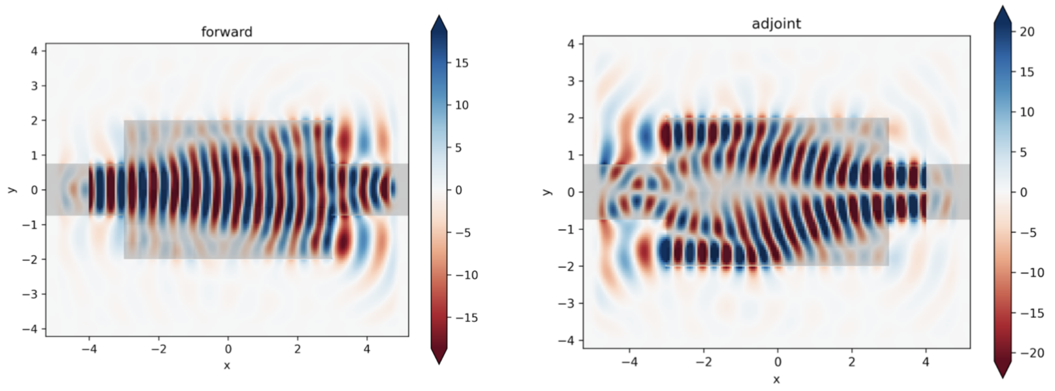
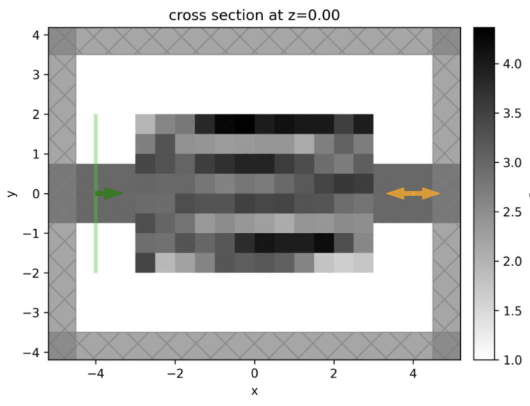
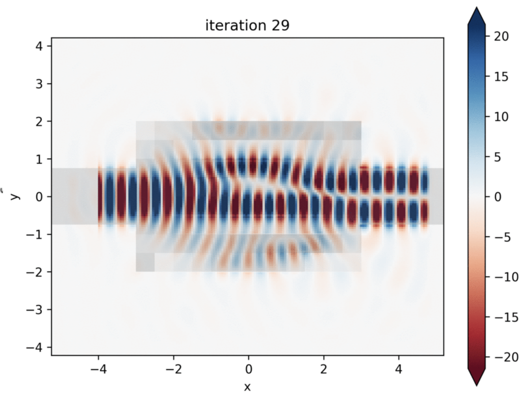


Figure 20: (left) The field pattern corresponding to the starting structure in normal operation when a fundamental mode is injected into the input port on the left-hand side. (b) The field pattern of the adjoint simulation of the same structure, in which the higher order mode is injected backwards into the simulation from the right-hand side output waveguide. By taking the overlap integral of these fields at the pixel locations, one may compute the gradient of the FoM with respect to the permittivities of each pixel using only two simulations per iteration.



(a)



(b)

Figure 21: (a) The relative permittivity of the device on its $z = 0$ cross section after 29 iterations of gradient-based optimization. (b) The field pattern associated with this device, showing almost perfect conversion from fundamental to first order mode, indicating a successful optimization of the FoM.

6 Conclusion

As we have shown, Tidy3D is a powerful and flexible FDTD simulator that offers orders of magnitude improvements in speed and scale. The combination of this performance with an easy to use interface and comprehensive set of features makes it an immensely valuable tool for photonic device design and electromangetics research. For more information on Tidy3D, we refer readers to the documentation and the front end code, which is freely available.

References

[1] Maitham Alhubail and Qiqi Wang. The swept rule for breaking the latency barrier in time advancing pdes. *Journal of Computational Physics*, 307:110–121, 2016.

[2] Antonios Giannopoulos. An improved new implementation of complex frequency shifted pml for the fdtd method. *IEEE Transactions on Antennas and Propagation*, 56(9):2995–3000, 2008.

[3] Minghui Han, Robert W Dutton, and Shanhui Fan. Model dispersive media in finite-difference time-domain method with complex-conjugate pole-residue pairs. *IEEE microwave and wireless components letters*, 16(3):119–121, 2006.

[4] Stephan Hoyer and Joe Hamman. xarray: Nd labeled arrays and datasets in python. *Journal of Open Research Software*, 5(1), 2017.

[5] Tyler W Hughes, Momchil Minkov, Victor Liu, Zongfu Yu, and Shanhui Fan. A perspective on the pathway toward full wave simulation of large area metalenses. *Applied Physics Letters*, 119(15):150502, 2021.

[6] Tyler W Hughes, Momchil Minkov, Ian AD Williamson, and Shanhui Fan. Adjoint method and inverse design for nonlinear nanophotonic devices. *ACS Photonics*, 5(12):4781–4787, 2018.

[7] John David Jackson. Classical electrodynamics, 1999.

[8] Po-Ru Loh, Ardavan F Oskooi, Mihai Ibanescu, Maksim Skorobogatiy, and Steven G Johnson. Fundamental relation between phase and group velocity, and application to the failure of perfectly matched layers in backward-wave structures. *Physical Review E*, 79(6):065601, 2009.

[9] Ahmad Mohammadi, Hamid Nadgaran, and Mario Agio. Contour-path effective permittivities for the two-dimensional finite-difference time-domain method. *Optics express*, 13(25):10367–10381, 2005.

[10] Sean Molesky, Zin Lin, Alexander Y Piggott, Weiliang Jin, Jelena Vucković, and Alejandro W Rodriguez. Inverse design in nanophotonics. *Nature Photonics*, 12(11):659–670, 2018.

[11] Ardavan F Oskooi, Lei Zhang, Yehuda Avniel, and Steven G Johnson. The failure of perfectly matched layers, and towards their redemption by adiabatic absorbers. *Optics Express*, 16(15):11376–11392, 2008.

[12] J Alan Roden and Stephen D Gedney. Convolution pml (cpml): An efficient fdtd implementation of the cfs–pml for arbitrary media. *Microwave and Optical Technology Letters*, 27(5):334–339, 2000.

[13] Allen Taflove, Susan C Hagness, and Melinda Piket-May. Computational electromagnetics: the finite-difference time-domain method. *The Electrical Engineering Handbook*, 3:629–670, 2005.

[14] Allen Taflove, Ardavan Oskooi, and Steven G Johnson. *Advances in FDTD computational electrodynamics: photonics and nanotechnology.* Artech House, 2013.

[15] Alexey Yamilov, Sergey E Skipetrov, Tyler W Hughes, Momchil Minkov, Zongfu Yu, and Hui Cao. Anderson localization of electromagnetic waves in three dimensions. *arXiv preprint arXiv:2203.02842*, 2022.

[16] Kane Yee. Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media. *IEEE Transactions on Antennas and Propagation*, 14(3):302–307, 1966.