

Transient Electron Dynamics Simulator

User Manual

v 1.2.

"Layers"

Contents

1	Introduction	2
2	Getting Started	3
2.1	Choosing a Simulation Model	3
2.2	Familiarizing with the Layout	4
3	The Inputs Tab - Defining the Model System	5
3.1	Overview	5
3.2	Laser Generation Conditions (LGC)	9
3.3	Parameter Toolkit	10
3.4	Parameter Toolkit Mathematical Methods	11
3.5	Parameter List Upload	18
3.6	Saving and Loading Model State Files	20
3.7	The Batch MSF Tool	21
3.8	Removing Files	22
4	Running Simulations	23
5	Analyzing and Integrating Simulation Data	25
5.1	Plotting	27
5.2	Stepping Through Time	29
5.3	Changing Axis Settings	29
5.4	Exporting Data	30
5.5	Regenerating MSFs from Existing Data	31
5.6	Integration	32
5.7	Time Series - Extracting Additional Information from Integrated Data	36
6	Adding a Custom Module - Python Familiarity Recommended	39
6.1	The Module Object Hierarchy	39
6.2	Filling Out the Module Template	39
6.2.1	__init__()	41
6.2.2	Defining Layers	41
6.2.3	Defining Parameters	43
6.3	calc_inits()	44
6.4	simulate()	44
6.5	get_overview_analysis()	44
6.6	prep_dataset()	44
6.7	get_timeseries()	45

6.8	<code>get_IC_carry()</code>	45
-----	---------------------------------------	----

1 Introduction

The Transient Electron Dynamics Simulator (TEDs) was originally designed to model a collection of time-dependent behaviors undergone by excited charge carriers within a nanowire with specified properties. Using a solver library like Scipy Integrate to evolve initial carrier distributions forward in time, the systems of differential equations can be solved numerically with a relatively simple Python script.

Not so simple, however, is adapting a single Python solver for many nanosystem variants. Editing parameters directly in source code requires relaunching the solver repeatedly. Manually constructing initial distributions is tedious. Any solver can get the job done, but a solver operating underneath the hood of a graphical interface opens many opportunities for utility features to the solver and restores some user friendliness by minimizing the need to interact directly with source code.

With that in mind, the goals of TEDs are to streamline how initial conditions and parameters are fed into the solver and offer more options for how to package the output of the solver. This we refer to as the initial state-simulation-data analysis workflow - a common pattern for many time-resolved finite-element problems.

TEDs has since expanded beyond the single problem it was originally designed for. To maximize re-usability of TEDs, any problem is defined as a standalone module file, a recipe detailing the parameters TEDs needs to apply, the distinct physical layers TEDs should track, the outputs TEDs needs to calculate, and any special instructions TEDs needs to observe. New and existing module files may be swapped in and out as needed.

2 Getting Started

2.1 Choosing a Simulation Model

When TEDs is launched, a Module Selection window appears:

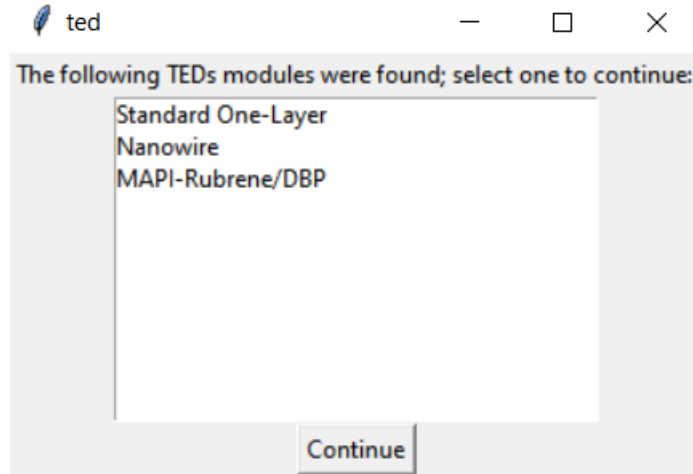


Figure 1: Three modules are available by default: a general-purpose one-layer material, a nanowire with axial rather than surface emission, and a specialized MAPI + Rubrene/DBP two-layer material.

TEDs scans the Modules directory for available module files and lists them in the Module Selection window. After selecting a module, TEDs performs some verification to ensure that the basic features of the module are correctly implemented, with all issues printed to console. This verification step does *not* check the accuracy of or detect problems in more advanced features, such as custom module functions, but TEDS will report any issues with these features at the time they are invoked. After verification, TEDs then checks for the presence of a set of working directories – “Initial”, “Data”, and “Analysis” – and automatically creates these in the same directory as the source code if they are missing. TEDs also checks for and creates a subdirectory within “Data” for the specific module being loaded.

It is *highly recommended* that all files created by TEDs remain in these directories – they are the only places TEDs will search for files, and TEDs may not be able to locate files stored elsewhere.

2.2 Familiarizing with the Layout

After choosing a module, an interface like the following is displayed:

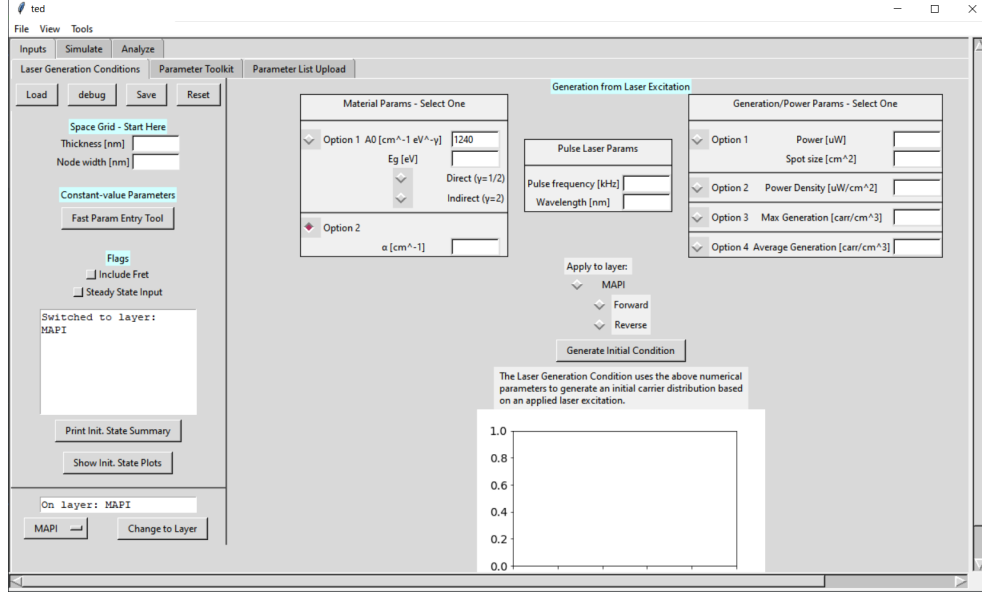


Figure 2: Main interface.

The top left corner contains a menu containing options to close the program, toggle fullscreen view, change the active module, and open various utility features not contained in the main interface.

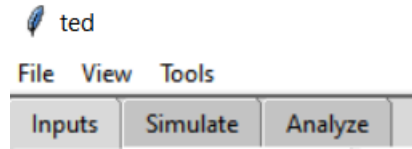


Figure 3: Main menu.

The top left corner also contains three tabs — Inputs, Simulate, and Analyze, which correspond to the three main simulation stages of preparing an initial condition and parameter set, simulating the evolution of the initial condition over time, and viewing and analyzing the results. Each of the sub-interfaces displayed by these tabs are described in the following sections.

3 The Inputs Tab - Defining the Model System

3.1 Overview

One of the most powerful features of TEDS is the ability to construct spatial distributions of *system parameters* in addition to *initial conditions*, allowing models whose properties differ positionally to be simulated with no more difficulty than homogeneous models. The first step of creating a model is to define a space grid - wherein TEDS partitions the model length into equally sized nodes of specified width. Each node center and node edge may then take on its own set of values independent of the others. These collections of values are then saved as model state files and later read by the simulator.

The Inputs Tab contains the following major components:

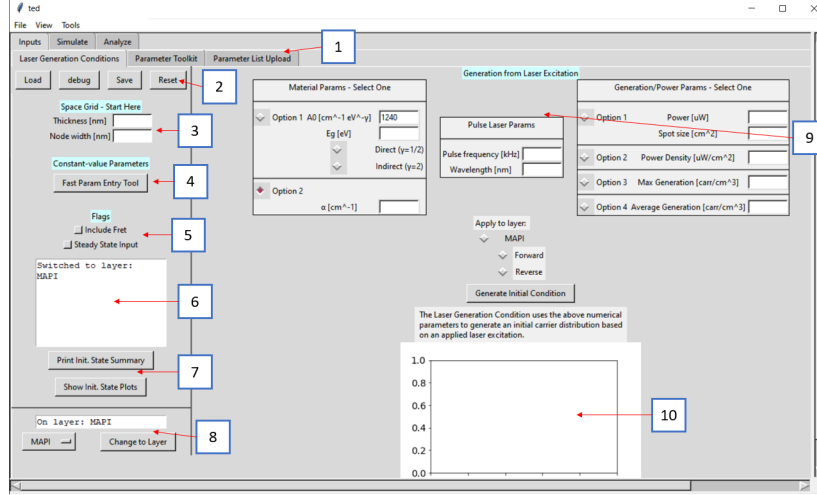


Figure 4: Key parts of Inputs Tab.

1. Initial Condition Subtabs

The Inputs Tab itself contains three subtabs – Laser Generation Condition, Parameter Toolkit, and Parameter List Upload. While initial condition and parameter distributions are zero by default, these three subtabs correspond to three methods available to modify them. A choice of subtab here affects the layout presented in [9].

2. Save, Load, and Reset Buttons

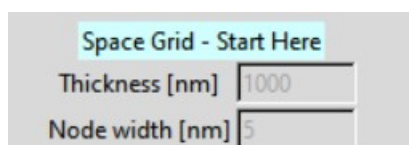
The Save and Load buttons are used to import and export model state files, discussed further in the sections *Importing Model State Files* and

Exporting Model State Files.

The Reset button clears specified system parameters and initial condition distributions from the interface and from the nodes, depending on the selected options, and is useful for correcting mistakes or restarting the model from a clean slate. Note that this feature does not affect the contents of previously saved model state files.

3. Space Grid Entry Area

Prior to entering node values, the space discretization grid for the model must first be established. The very first step of creating an model state file is entering the total length and node width (dx) here.



The image shows a graphical user interface window titled "Space Grid - Start Here". Inside the window, there are two labeled input fields. The first field is labeled "Thickness [nm]" and contains the numerical value "1000". The second field is labeled "Node width [nm]" and contains the numerical value "5". The fields are simple rectangular boxes with a thin border.

Figure 5: A "locked in" space grid of length 1000 nm and node width 5 nm. This grid consists of 200 nodes whose centers are at 2.5, 7.5, 12.5...997.5 nm and whose edges are at 0, 5, 10...1000 nm.

Once any node value is assigned, the space grid becomes "locked in." To enter a new space grid, clear the existing space grid using the "Clear All" option in the Reset button window.

4. Fast Parameter Entry Tool

Oftentimes, parameters will be constant throughout the entire system. This tool can be used to enter constant valued parameters quickly, while other tools should be used for more detailed distributions.

Parameter Short-cut Tool				Are the values of certain parameters constant across the system? Enter those values here and press "Continue" to apply them on all space grid points.			
mu_N [cm ² / V s]	20	tau_P [ns]	20	back_reflectivity	0	Ec [eV]	0
mu_P [cm ² / V s]	20	Sf [cm / s]	10	alpha [cm ⁻¹]	0	electron_affinity [eV]	0
N0 [carr / cm ³]	1e8	Sb [cm / s]	10	delta	0		
P0 [carr / cm ³]	1e15	temperature [K]	300	frac_emitted	1		
B [cm ³ / s]	1e-10	rel_permitivity	13.6	delta_N [carr / cm ³]	0		
tau_N [ns]	20	Ext_E-Field [V/um]	0	delta_P [carr / cm ³]	0		
Continue							

Figure 6: Example list of values for each of the standard one-layer model's parameters. When "Continue" is clicked, all nodes and edges are assigned these values.

Parameters may be entered as numbers – e.g. 100, 9999, or 0.25 – or in scientific notation – e.g. 1e2, 9.999e3, 2.5e-1

5. Flags Area

Flags

☐ Ignore Photon Recycle

☐ Steady State Input

Figure 7: Flags for the One-Layer module.

Module-specific flags can be toggled here. By default, each of the three included modules has two flags.

The "Ignore Photon Recycle" flag deactivates carrier regeneration and photon recycling effects within the simulation. If these effects are known to be minor, neglecting these terms can speed up the simulation.

The "Include FRET" flag determines whether fretting is active in the MAPI-Rubrene/DBP simulation. As with the "Ignore Photon Recycle" flag, this can be deactivated for a speedup if fretting is negligible.

The "Steady State Input" flag alters how the initial carrier densities ΔN and ΔP are treated. When active, TEDs assumes that contin-

uously, ΔN and ΔP carriers per nm^{-3} are added to the model per ns . This also alters how the Laser Generation Condition works - the LGC calculates a carrier injection rate ($\frac{carriers}{nm^3 ns}$) rather than an initial carrier density distribution ($\frac{carriers}{nm^3}$) as from a single laser pulse.

Furthermore, the Nanowire module defines an always-active hidden “symmetric system” flag – as according to its experimental setup (i.e. the Nanowire is excited at its center whereas the one-layer is excited at its surface). Any module, (including user-made ones) which includes this flag will, when this flag is activated for a model of length L , calculate a mirror model over the range 0 to $-L$ by copying the values from the actual model spanning 0 to L . This primarily affects whether photon recycle effects from the mirror space are included and whether integration into the mirror space is allowed.

6. Status Box

This box displays status and error messages associated with managing the model state.

7. Model Summary Buttons

These buttons can be used to view a list or plots of the current parameter values and can be used to verify a model state before saving to a file.

8. Layer Selection Window

This area is used to swap between a module’s layers. Following their namesakes, the OneLayer and Nanowire modules feature one layer while the MAPI-Rubrene/DBP module features one each for MAPI and Rubrene/DBP.

Each layer maintains its own space grid and set of parameters. Check which layer is currently active in the status box to ensure that changes are being made to the correct layer.

9. Parameter Distribution Generation Suite

The layout of this area and options available changes depending on which tab of [1] is selected. Details regarding how each of the generation suites operate are available in the subsections *Laser Generation Conditions*, *Parameter Toolkit*, and *Parameter List Upload*.

10. Plots

Each plot appearing in this area presents a visualization of recently updated parameters of the initial state.

Toolbars below offer resizing and scaling of the Initial State Plots as well as options to export images of the plots.

3.2 Laser Generation Conditions (LGC)

The screenshot shows a software interface for setting laser generation conditions. It includes input fields for material parameters (A0, Eg, alpha), pulse parameters (frequency, wavelength), and generation/power parameters (Power, Power Density, Max Generation, Average Generation). A central button 'Generate Initial Condition' is present, along with a descriptive text box and a plot area.

Figure 8: Laser Generation Condition Tab.

For select modules, the Laser Generation Condition can be used to generate ΔN and ΔP according to a series of laser pulse excitation equations.

Using the laser parameters shown above, this mode assigns values to ΔN and ΔP at each node for the selected layer using the following equations:

First, the absorption coefficient α may be calculated from the below or entered directly.

$$\alpha = A_0 \left(\frac{hc}{\lambda} - E_g \right)^\gamma \quad (1)$$

The laser pulse profile may then be generated from one of four equation forms:

$$\Delta N(z) = \Delta P(z) = \frac{W}{Af \left(\frac{hc}{\lambda} \right)} \alpha e^{-\alpha z} \quad (2)$$

$$\Delta N(z) = \Delta P(z) = \frac{w''}{f\left(\frac{hc}{\lambda}\right)} \alpha e^{-\alpha z}, w'' = \frac{W}{A} \quad (3)$$

$$\Delta N(z) = \Delta P(z) = G_{max} e^{-\alpha z} \quad (4)$$

$$\Delta N(z) = \Delta P(z) = G_{avg} L \frac{\alpha e^{-\alpha L}}{e^{-\alpha z} (e^{-\alpha L} - 1)} \quad (5)$$

TEDs automatically performs all unit conversions needed to obtain ΔN and ΔP in $\frac{\text{carriers}}{\text{nm}^3}$ from values inputted in the listed units.

3.3 Parameter Toolkit

The Parameter Toolkit offers a fast, versatile way to sketch initial condition profiles by inputting a list of mathematical rules. Key parts of this interface are:

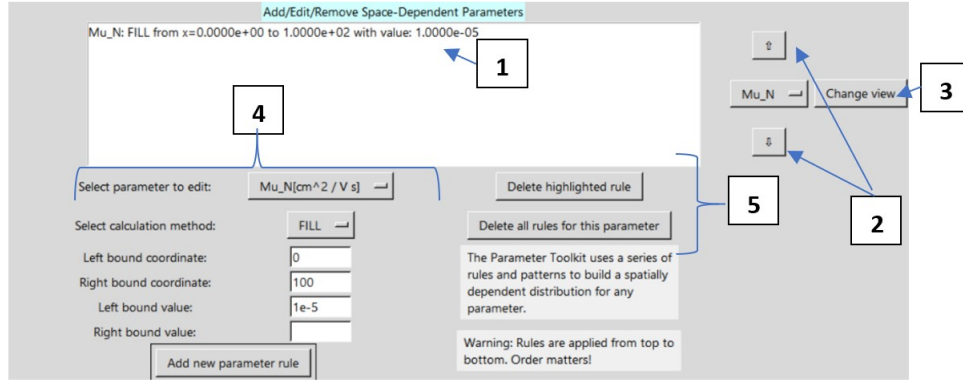


Figure 9: Parameter toolkit with one rule entered for parameter μ_n . This rule, for example, assigns the value $10^{-5} \frac{\text{cm}^2}{\text{Vs}}$ to all nodes spanning 0 nm to 100 nm.

1. Rule List Window

When a new parameter rule is added, a brief description of it will be added to this window. TEDs applies each of these rules one at a time, in order from top to bottom, to construct space distributions for the indicated parameter. Only one parameter's rules are displayed at a time – see [3] for changing which parameter is displayed.

2. Reordering Buttons

Rules in [1] can be selected with a click and can be reordered in the list using these buttons. The order of which rules appear in the list is the order in which TEDs will apply them to the distribution - be sure to view the plots to ensure that the distribution is constructed as desired.

3. Parameter Selection

Select a parameter using the drop-down menu and click “Change view” to display that parameter’s rule list in [1]. This can also be used to change which parameter is plotted in the “Snapshot” plot.

4. Rule Input Boxes

These items are used to, for a new rule, specify the parameter for which it should be applied, the mathematical method used to apply it, and the values that the method should use. There are four mathematical methods – POINT, FILL, LINE, and EXP, and each of these are discussed in the following section *Parameter Toolkit Mathematical Methods*.

5. Deletion Buttons

These buttons are used to delete either the selected rule or clear all rules for the currently displayed parameter.

3.4 Parameter Toolkit Mathematical Methods

Four mathematical methods are available for the parameter toolkit – POINT, FILL, LINE, and EXP.

The POINT method is by far the most straightforward, assigning the Left Bound Value to the selected parameter’s initial distribution at the location specified by Left Bound Coordinate. Generally this refers to the single node containing that location, so the resolution of this method is specified by the node width. In the following example, this method is applied twice to create a very pointy ΔP initial distribution.

Add/Edit/Remove Space-Dependent Parameters

deltaP: POINT at x=2.0000e+03 with value: 1.0000e+18
 deltaP: POINT at x=6.0000e+03 with value: 3.0000e+17

Select parameter to edit: deltaP[cm⁻³]

Select calculation method: POINT

Left bound coordinate: 6000

Right bound coordinate:

Left bound value: 3e17

Right bound value:

Add new parameter rule

Delete highlighted rule

Delete all rules for this parameter

The Parameter Toolkit uses a series of rules and patterns to build a spatially dependent distribution for any parameter.

Warning: Rules are applied from top to bottom. Order matters!

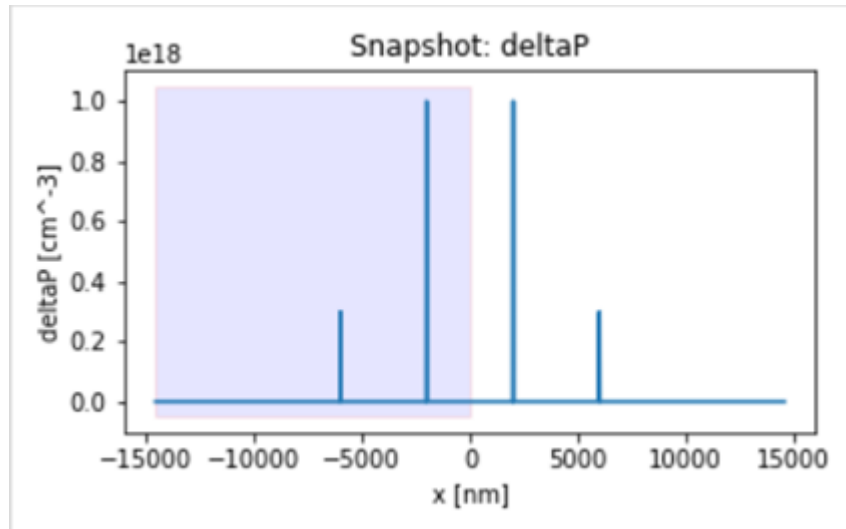


Figure 10: In this example with the Nanowire module, the symmetric system flag is active - note the mirrored half of the system represented in the plot.

The FILL method fills every space point from the Left Bound Coordinate to the Right Bound Coordinate with the Left Bound Value. This method is useful for specifying regions with constant values and is a special case of the LINE method.

Add/Edit/Remove Space-Dependent Parameters

Mu_N: FILL from x=1.0000e+03 to 2.0000e+03 with value: 2.5000e+01
 Mu_N: FILL from x=2.0000e+03 to 4.0000e+03 with value: 1.2000e+01
 Mu_N: FILL from x=6.0000e+03 to 9.0000e+03 with value: 1.0000e+01

Select parameter to edit: Mu_N[cm² / V s]

Select calculation method: FILL

Left bound coordinate: 6000

Right bound coordinate: 9000

Left bound value: 10

Right bound value:

Add new parameter rule

Delete highlighted rule

Delete all rules for this parameter

The Parameter Toolkit uses a series of rules and patterns to build a spatially dependent distribution for any parameter.

Warning: Rules are applied from top to bottom. Order matters!

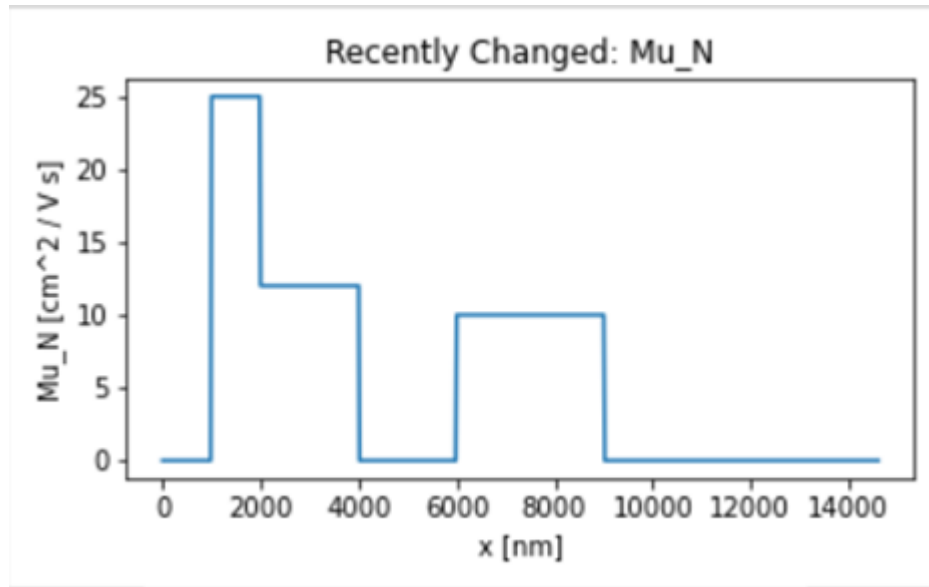


Figure 11: In this example, three FILL commands are used to create a staircase-shaped distribution.

The LINE method assigns the Left Bound Value to the Left Bound Coordinate, assigns the Right Bound Value to the Right Bound Coordinate, and performs linear interpolation to assign values to all intermediate space

points. Each intermediate point differs from its neighbors by a common difference value.

Add/Edit/Remove Space-Dependent Parameters

Mu_N: LINE from x=0.0000e+00 to 1.0000e+02 with left value: 1.0000e+00 and right value: 1.0000e+01

Select parameter to edit: Mu_N[cm^2 / V s]

Delete highlighted rule

Select calculation method: LINE

Delete all rules for this parameter

Left bound coordinate: 0

Right bound coordinate: 100

Left bound value: 1

Right bound value: 10

Add new parameter rule

The Parameter Toolkit uses a series of rules and patterns to build a spatially dependent distribution for any parameter.

Warning: Rules are applied from top to bottom. Order matters!

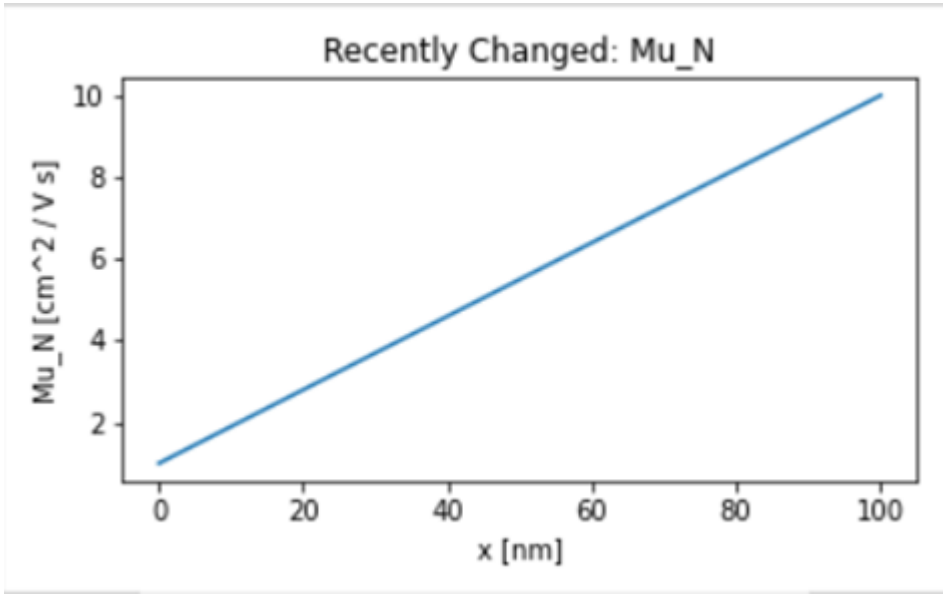


Figure 12: In this example, the LINE command is used to draw a distribution from (0 nm, 1) to (100 nm, 10).

The EXP method is similar to LINE, but intermediate space points are instead filled by exponential interpolation between the left and right bounds. Each intermediate point differs from its neighbors by a common ratio.

Add/Edit/Remove Space-Dependent Parameters

Mu_N: EXP from x=0.0000e+00 to 8.0000e+01 with left value: 1.0000e+05 and right value: 1.0000e+01

Select parameter to edit:
Mu_N[cm² / V s]
Delete highlighted rule

Select calculation method:
EXP
Delete all rules for this parameter

Left bound coordinate:
0

Right bound coordinate:
80

Left bound value:
1e5

Right bound value:
1e1

Add new parameter rule

The Parameter Toolkit uses a series of rules and patterns to build a spatially dependent distribution for any parameter.

Warning: Rules are applied from top to bottom. Order matters!

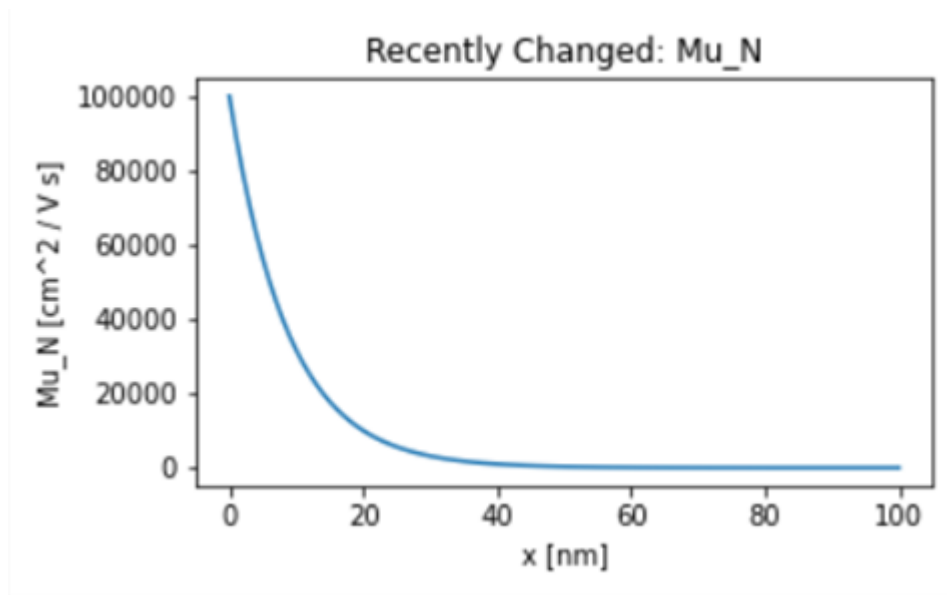


Figure 13: "In this example, an exponential curve is drawn from (0 nm, 100 000) to (80 nm, 10). The remaining space from 80 to 100 nm has value zero.

These four methods can be combined to set up complex distributions.

Add/Edit/Remove Space-Dependent Parameters

Mu_N: FILL from x=0.0000e+00 to 1.5000e+03 with value: 1.0000e+02
 Mu_N: LINE from x=3.0000e+02 to 5.0000e+02 with left value: 1.0000e+02 and right value: 5.0000e+02
 Mu_N: LINE from x=0.0000e+00 to 2.0000e+02 with left value: 1.0000e+02 and right value: 5.0000e+02
 Mu_N: LINE from x=6.0000e+02 to 8.0000e+02 with left value: 1.0000e+02 and right value: 5.0000e+02
 Mu_N: LINE from x=9.0000e+02 to 1.1000e+03 with left value: 1.0000e+02 and right value: 5.0000e+02
 Mu_N: LINE from x=1.2000e+03 to 1.4000e+03 with left value: 1.0000e+02 and right value: 5.0000e+02

Select parameter to edit: Mu_N[cm² / V s]

Select calculation method: LINE

Left bound coordinate: 1200

Right bound coordinate: 1400

Left bound value: 100

Right bound value: 500

Add new parameter rule

Delete highlighted rule

Delete all rules for this parameter

The Parameter Toolkit uses a series of rules and patterns to build a spatially dependent distribution for any parameter.

Warning: Rules are applied from top to bottom. Order matters!

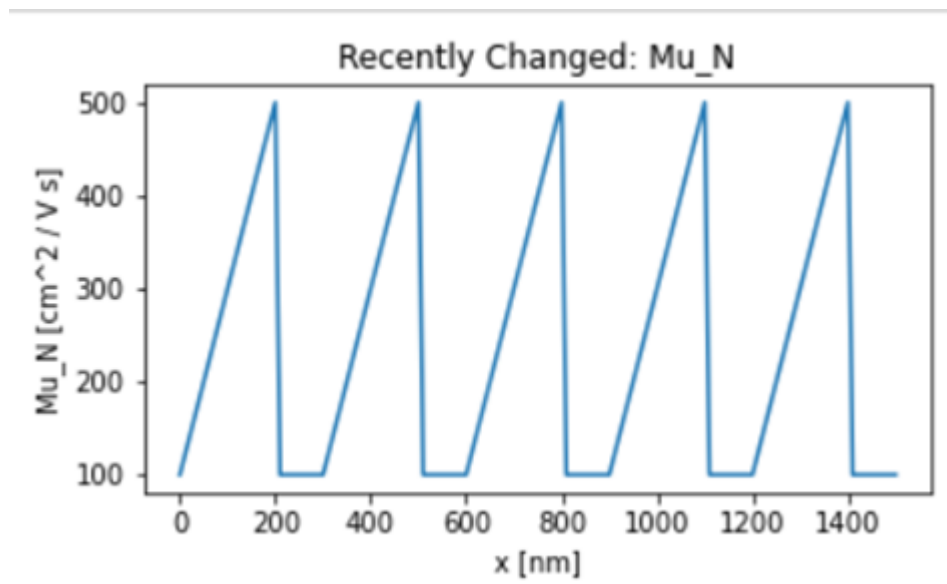


Figure 14: LINE and FILL methods combined to create a "sawtooth" distribution.

3.5 Parameter List Upload

Finally, TEDs can accept custom distributions in the form of .txt files containing lists of space coordinate and value pairs and apply these to a selected variable. Any space coordinates between those specified in the file will be filled in by linear interpolation. These .txt files must be formatted as two tab-separated columns, with the first column for coordinates and the second column for values

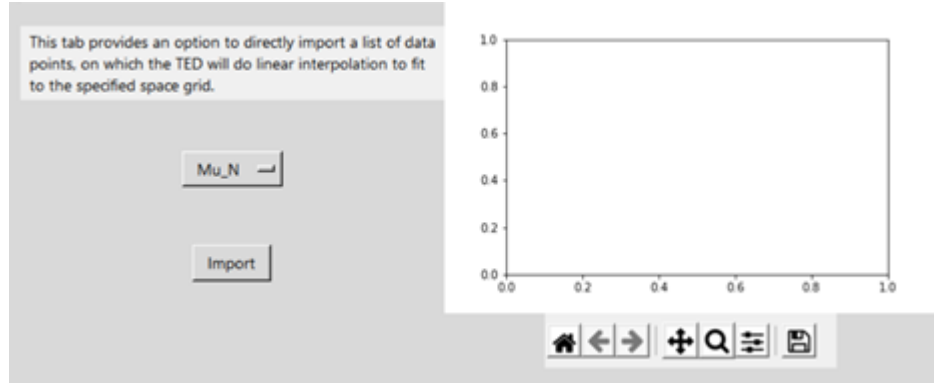


Figure 15: List upload tab.

In the following example, the file “points.txt” is applied to ΔN over the range $z=0$ nm to $z=6000$ nm.

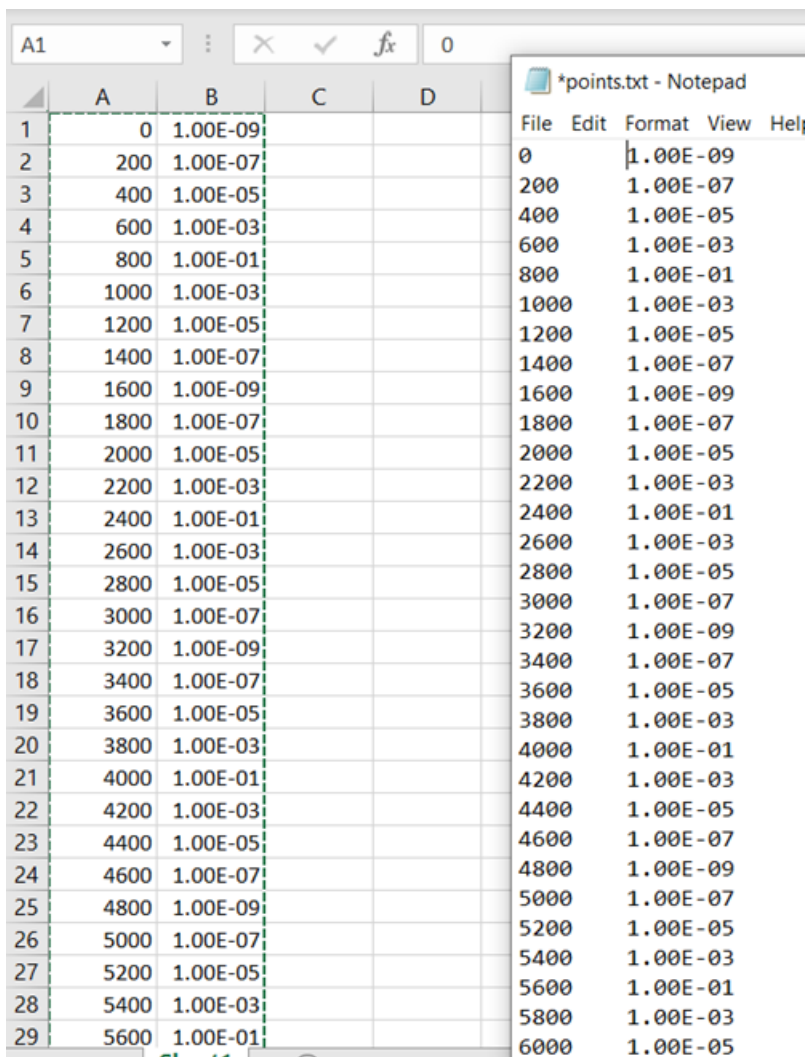


Figure 16: Copying data from a spreadsheet program such as Excel can automatically provide the necessary tab-separated format.

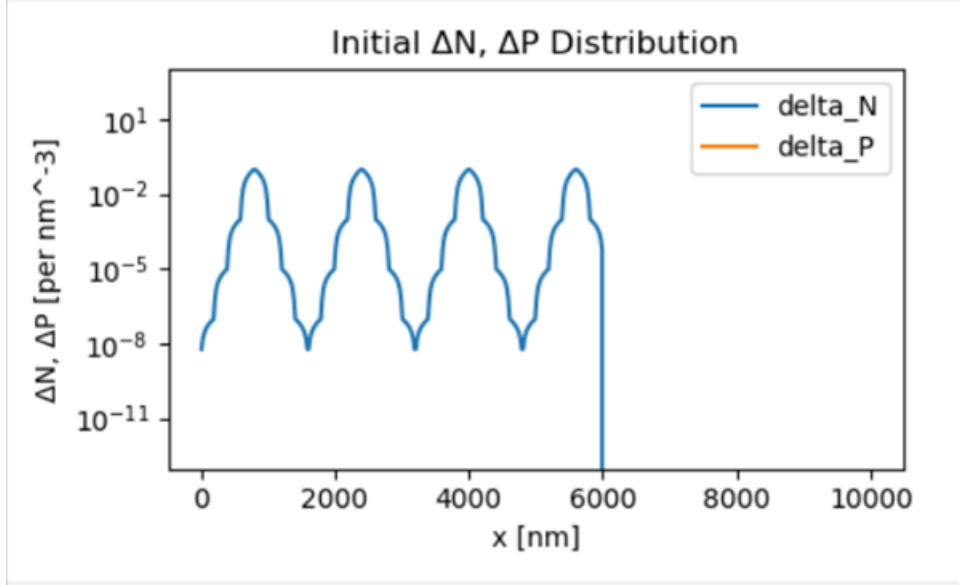


Figure 17: The data are then linear interpolated to produce a bumpy-looking distribution on a log-scale plot up to 6000 nm, the final point in the file.

3.6 Saving and Loading Model State Files

Before TEDs can simulate from initial distributions and system parameter sets, these must be saved as model state files (MSFs) using the “Save” feature.



Figure 18: Also included is a debug button - this prints out log messages but serves no other function.

Once all system parameters have been entered, clicking the “Save” button will allow you to create and name a new model state file. MSFs have a specific layout designed to inform TEDs of where different items are located in the file.

Saving an MSF without all system parameters entered is possible. In that case, all of the distributions would be saved as their default values – zeroes.

3.7 The Batch MSF Tool

In many situations, such as sensitivity analyses, it is useful to generate many model state files over a rectangular parameter space. In the “Tools” menu, TEDs offers the Batch Model State File Tool, which provides a fast method to generate many such copies based on an existing initial state.

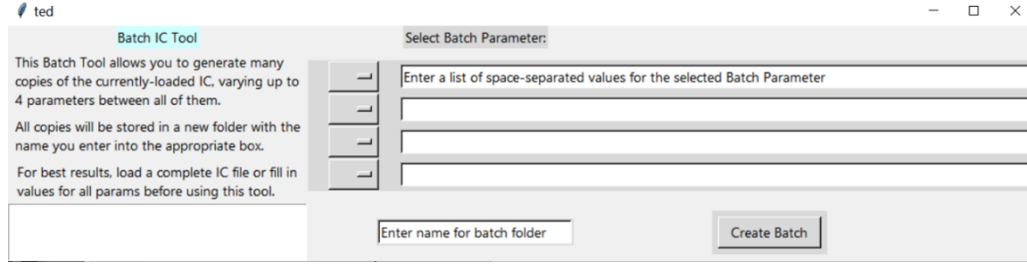


Figure 19: Batch MSF Tool when first opened.

In the following example, the Batch MSF Tool is used to generate copies of “2-9-21.txt” from *Saving and Loading Initial Conditions* with varying values of the parameter τ_N : 10, 25, 40, 100, 1 000, and 10 000.

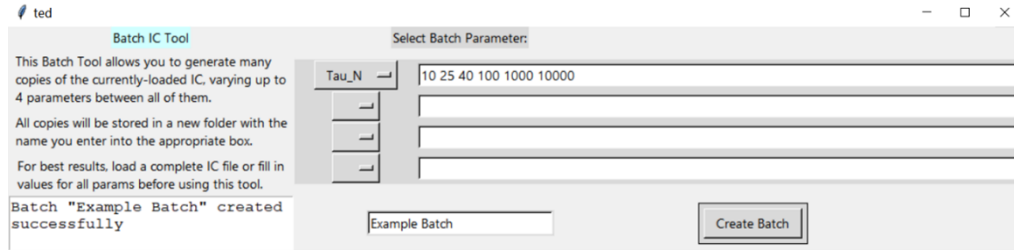
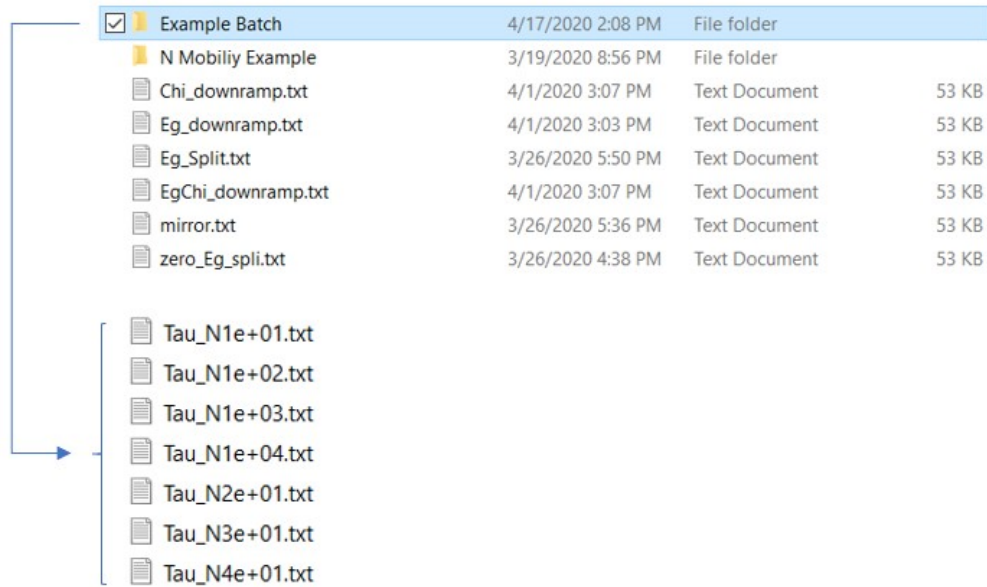


Figure 20: Batch tool with example values entered.

When “Create Batch” is clicked, a directory is created within the Initial Directory, alongside any existing MSFs. Each file in the batch is procedurally named using the values of the varied parameters it has adopted.



File Name	Modified Date	File Type	Size
Example Batch	4/17/2020 2:08 PM	File folder	
N Mobiliy Example	3/19/2020 8:56 PM	File folder	
Chi_downramp.txt	4/1/2020 3:07 PM	Text Document	53 KB
Eg_downramp.txt	4/1/2020 3:03 PM	Text Document	53 KB
Eg_Split.txt	3/26/2020 5:50 PM	Text Document	53 KB
EgChi_downramp.txt	4/1/2020 3:07 PM	Text Document	53 KB
mirror.txt	3/26/2020 5:36 PM	Text Document	53 KB
zero_Eg_spli.txt	3/26/2020 4:38 PM	Text Document	53 KB
Tau_N1e+01.txt			
Tau_N1e+02.txt			
Tau_N1e+03.txt			
Tau_N1e+04.txt			
Tau_N2e+01.txt			
Tau_N3e+01.txt			
Tau_N4e+01.txt			

Figure 21: Resulting batch of MSF files.

Up to four parameters may be varied at a time.

3.8 Removing Files

In the top left corner menu under “File”, TEDs provides a series of shortcuts to each of the working directories.

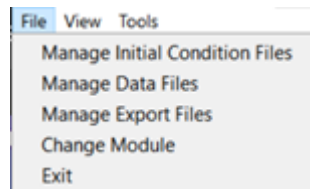


Figure 22: Clicking the File menu reveals various management options for the overall program.

Each of these opens the corresponding working directory, from where any files created by TEDs can be deleted.

4 Running Simulations

The Simulate tab contains the following major components:

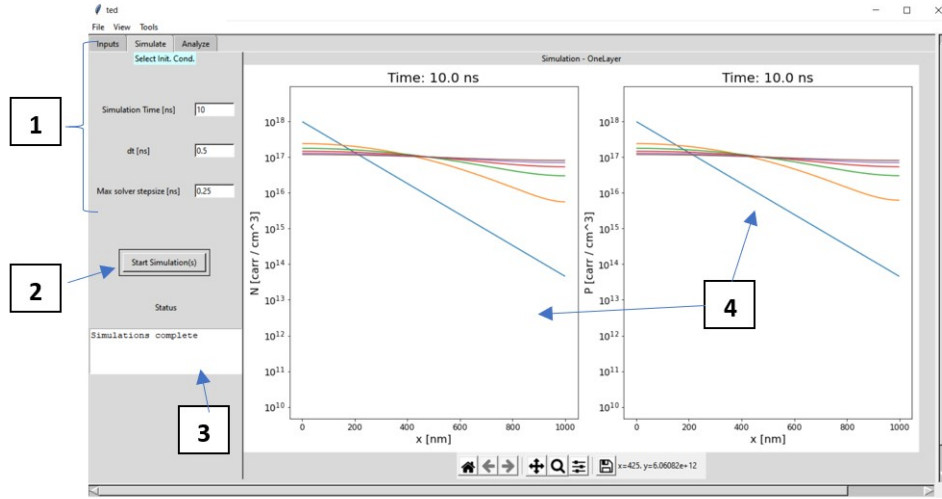


Figure 23: Key features of the Simulate Tab.

1. Simulation Parameters

This area contains input boxes in which the total time over which the system should be modeled and the time step size (dt) into which this total should be partitioned. Options to modify the behavior of the simulation, such as module-specific flags, are specified per MSF in the Initial tab.

There is also an input box to set the maximum time stepsize taken by solvers such as ODEINT. If left blank, by default the solver attempts to select an optimal internal stepsize to maximize speed while interpolating the data to the actual requested stepsize. However, sometimes this affects the accuracy of the solutions. If more accuracy is desired, a maximum stepsize can be specified here.

For general cases, the recommended dt and solver stepsizes are shown above.

2. Calculate Button

When this button is clicked, a prompt opens for selecting previously saved MSFs. Multiple files may be selected and TEDs will simulate

each of these in series with the settings in [1].

When a simulation is complete, TEDs will create a folder containing the results of the simulation in the corresponding module subdirectory within the “Data” directory. The name of this folder is based on the name of the MSF used to run the simulation.

3. Status Window

Like its counterpart in the inputs tab, this window displays status messages and problems encountered when performing the simulations.

4. Results Windows

These plots display snapshots of time steps taken during the simulation, a useful first glance of how the model is evolving over time. As with the Initial tab plots, these plots have a toolbar for basic resizing and exporting.

5 Analyzing and Integrating Simulation Data

The Analyze tab contains two subtabs – Overview and Detailed Analysis.

The Overview tab plots a snapshot of various outputs over time for a selected data folder. For the One-Layer and Nanowire modules, these are, in addition to ΔN and ΔP : the internal electric field, nonradiative and radiative combination, TRPL over full thickness, and effective lifetime τ_{diff}

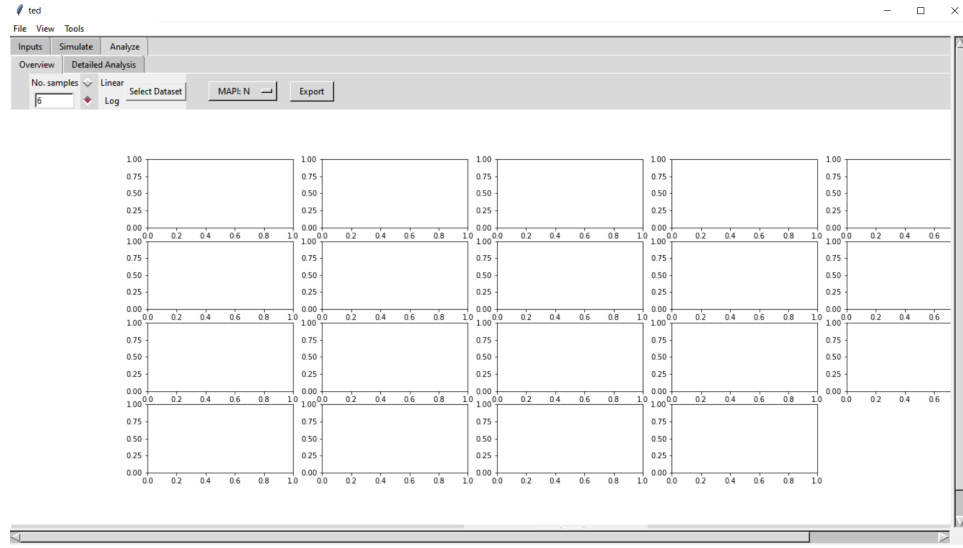


Figure 24: The Overview subtab when first opened. Click "Select Dataset" to choose a data set.

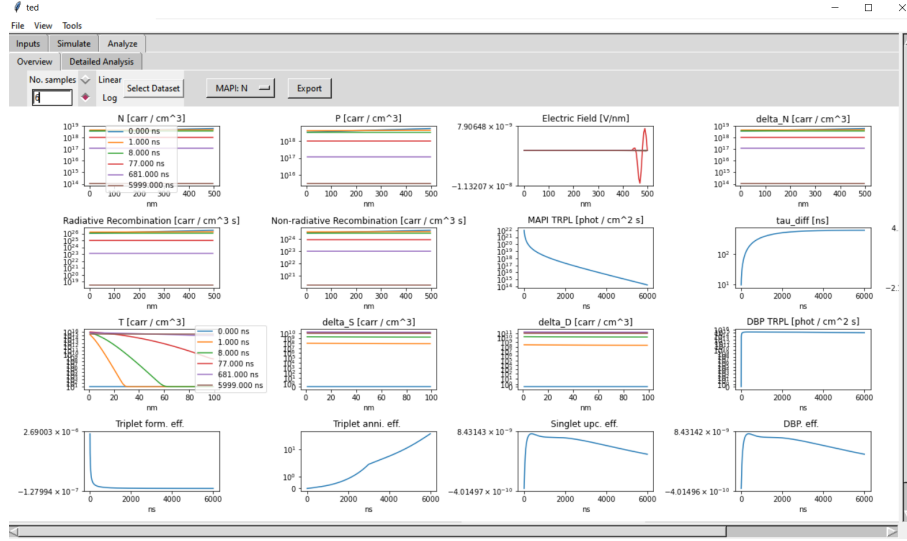


Figure 25: An example overview of a MAPI/Rubrene-DBP data set.

Two types of plots are displayed - plots which display selected timesteps over the space grid and plots which display values calculated over the time grid. The "No. Samples" box and adjacent log/linear selector determine which timesteps are selected.

To view other timesteps or TRPL over other ranges, for example, the Detailed Analysis tab should be used.

The Detailed Analysis tab contains four smaller plots – for navigating through the time steps of selected data sets, and one larger plot – for showing results from integrating data sets from the smaller plots.

The defining feature of this tab is that each plot is equipped with a more detailed toolbar, each feature of which is covered in the following sections.

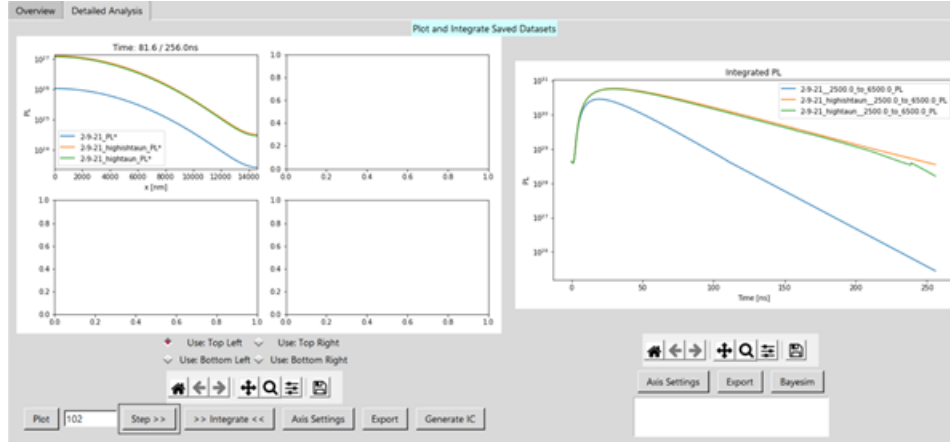


Figure 26: An example of the Detailed Analysis tab in use. Here the TRPL data of three datasets is compared.



Figure 27: Toolbar for analysis plots.

5.1 Plotting

The “Plot” button opens the following popup, from which there are options to select which variable should be plotted on the y-axis and which data sets should be plotted.

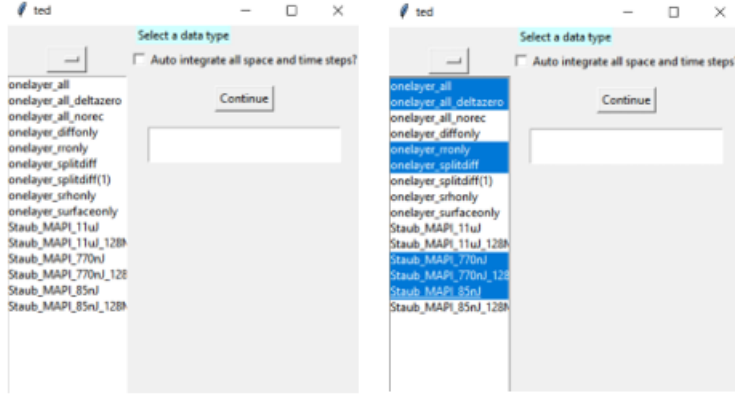


Figure 28: To select a single dataset, click its name in the box. To select multiple datasets, hold the Ctrl key and drag the cursor over every desired dataset.

When “Continue and select datasets” is clicked, the first time step of each selected dataset (i.e. the initial condition) is plotted. If the “Auto Integrate All Time and Space Steps” option is selected, TEDs will also integrate all time steps over the full length of the system and display the result in the large plot.

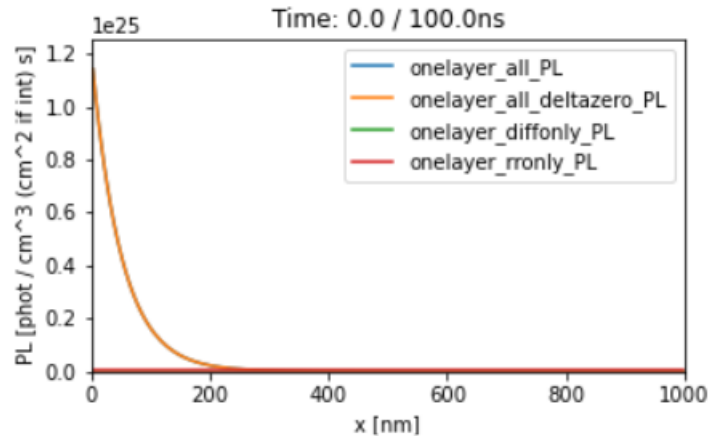


Figure 29: PL data from the previously selected data sets. All data when first plotted show their initial $t=0$ values.

One limitation, as demonstrated above by the fact that only four of the seven selected datasets have been plotted, is that the plotter can only plot datasets together if they have the same total time and time step size. If datasets with different total time or time step size should be compared, plotting each on a different plot is recommended.

Datasets with different space grids and identical time grids, however, are compatible with one another.

5.2 Stepping Through Time

Directly to the right of the “Plot” button is the “Step” input box and button. When the “Step” button is clicked, all plotted datasets are advanced by the number of time steps specified in the input box. How far ahead the datasets are advanced in absolute time depends on the size of the time steps used in the simulation.

With a time step size of 0.5 ns, for example, entering 20 will advance the datasets by 10.0 ns. each time the “Step” button is clicked.

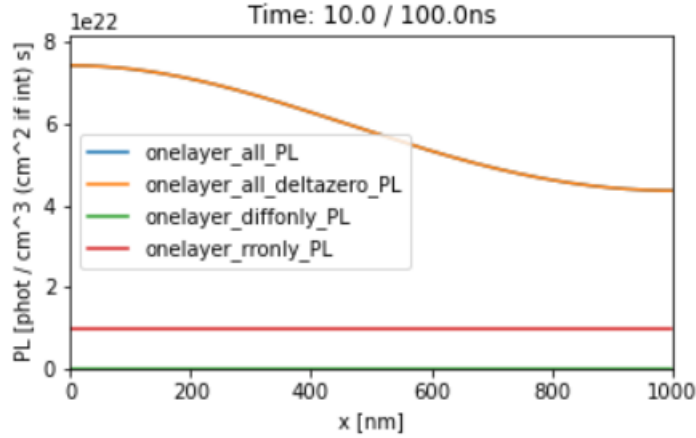


Figure 30: Data sets at a later time.

Stepping with a negative number of time steps will cause TEDs to move the datasets *backward* that many time steps.

5.3 Changing Axis Settings

The “Axis Settings” button opens a popup to change the lower bounds, upper bounds, and scaling type of the x and y axes. Options to toggle the legend

visibility or freeze the axes are also available.

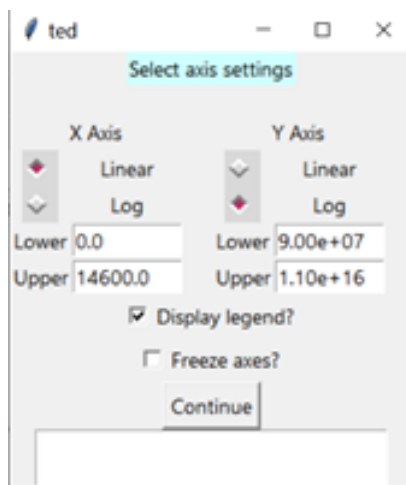


Figure 31: Adjusting the x and y axes. When opened, the Axis Settings popup is populated with the current axis values.

TEDs will attempt to determine the best axis range and scale (linear or log) based on the values spanned by the data and will update the axes continuously as the displayed data is changed. Sometimes this does not work well, and other times static axes are preferred. In that case the “Freeze Axes” option can be used to fix and specify the axes manually.

5.4 Exporting Data

The “Export” button opens a prompt to save the currently plotted data to a .csv file, which can be viewed with Excel or any text editor. These files have an alternating column format – where x is the x-axis variable (position, time, or parameter value) and y is the y-axis variable, the first and second columns are the (x, y) data points of the first data set, the third and fourth columns are (x, y) for the second data set, and so on.

	A	B	C	D	E	F
1	# x [nm]	onelayer_all	x [nm]	onelayer_all_deltazero	x [nm]	onelayer_diffonly
2	2.50E+00	7.44E+22	2.50E+00	7.44E+22	2.50E+00	0.00E+00
3	7.50E+00	7.44E+22	7.50E+00	7.44E+22	7.50E+00	0.00E+00
4	1.25E+01	7.43E+22	1.25E+01	7.44E+22	1.25E+01	0.00E+00
5	1.75E+01	7.43E+22	1.75E+01	7.44E+22	1.75E+01	0.00E+00
6	2.25E+01	7.43E+22	2.25E+01	7.43E+22	2.25E+01	0.00E+00
7	2.75E+01	7.43E+22	2.75E+01	7.43E+22	2.75E+01	0.00E+00
8	3.25E+01	7.43E+22	3.25E+01	7.43E+22	3.25E+01	0.00E+00
9	3.75E+01	7.42E+22	3.75E+01	7.43E+22	3.75E+01	0.00E+00
10	4.25E+01	7.42E+22	4.25E+01	7.42E+22	4.25E+01	0.00E+00
11	4.75E+01	7.42E+22	4.75E+01	7.42E+22	4.75E+01	0.00E+00
12	5.25E+01	7.41E+22	5.25E+01	7.41E+22	5.25E+01	0.00E+00
13	5.75E+01	7.41E+22	5.75E+01	7.41E+22	5.75E+01	0.00E+00
14	6.25E+01	7.40E+22	6.25E+01	7.40E+22	6.25E+01	0.00E+00
15	6.75E+01	7.40E+22	6.75E+01	7.40E+22	6.75E+01	0.00E+00
16	7.25E+01	7.39E+22	7.25E+01	7.39E+22	7.25E+01	0.00E+00
17	7.75E+01	7.39E+22	7.75E+01	7.39E+22	7.75E+01	0.00E+00
18	8.25E+01	7.38E+22	8.25E+01	7.38E+22	8.25E+01	0.00E+00
19	8.75E+01	7.37E+22	8.75E+01	7.37E+22	8.75E+01	0.00E+00
20	9.25E+01	7.36E+22	9.25E+01	7.37E+22	9.25E+01	0.00E+00
21	9.75E+01	7.36E+22	9.75E+01	7.36E+22	9.75E+01	0.00E+00
22	1.03E+02	7.35E+22	1.03E+02	7.35E+22	1.03E+02	0.00E+00
23	1.08E+02	7.34E+22	1.08E+02	7.34E+22	1.08E+02	0.00E+00
24	1.13E+02	7.33E+22	1.13E+02	7.33E+22	1.13E+02	0.00E+00
25	1.18E+02	7.32E+22	1.18E+02	7.32E+22	1.18E+02	0.00E+00

Figure 32: Example exported .csv file viewed using Microsoft Excel.

5.5 Regenerating MSFs from Existing Data

Finally, the “Generate IC” button opens a popup that can be used to construct a new MSF from previously simulated datasets.

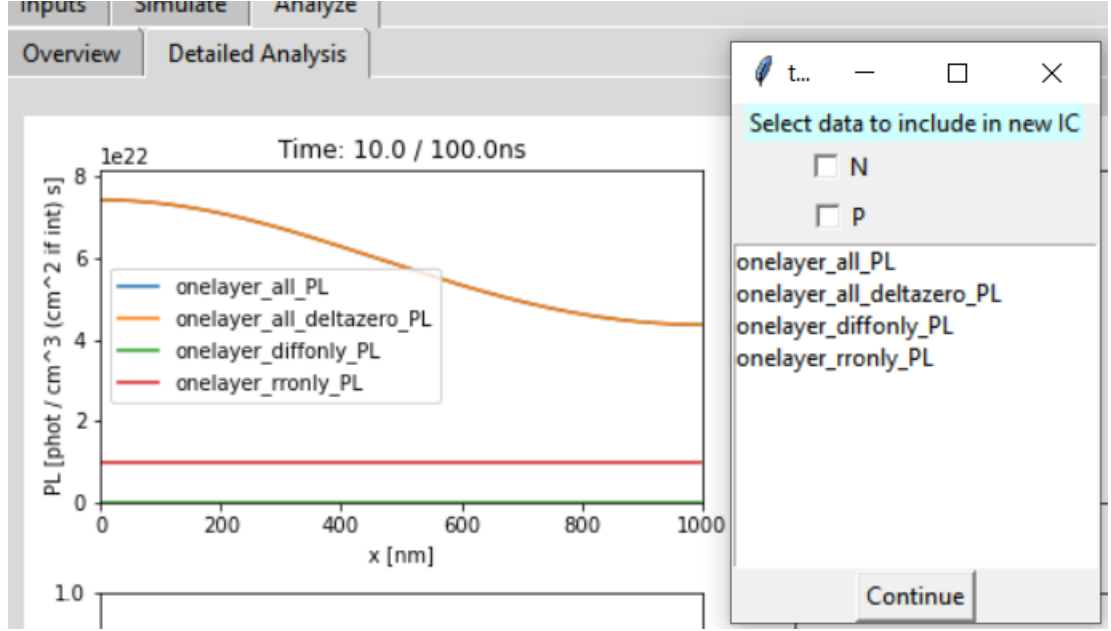


Figure 33: Example MSF regeneration popup based on the currently loaded data sets.

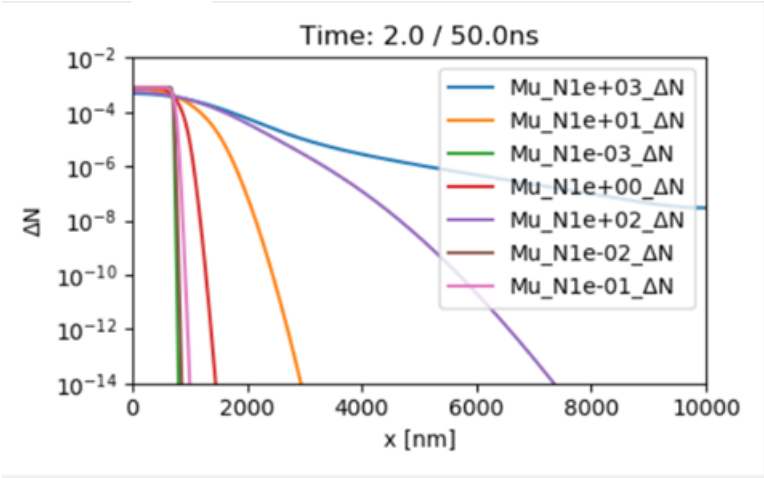
First, the checkboxes are used to indicate which variables should be copied into the new MSFs. For the one-layer and nanowire modules, ΔN and ΔP may be copied. The selection box, which functions like the “Plot” button’s, is then used to indicate which datasets MSFs should be generated for. When “Continue” is clicked, one MSF will be created with the selected variables for each dataset and prompts will appear to name the MSFs.

These MSFs can then be modified further using the Initial tab or simulated using TEDs’ other tabs.

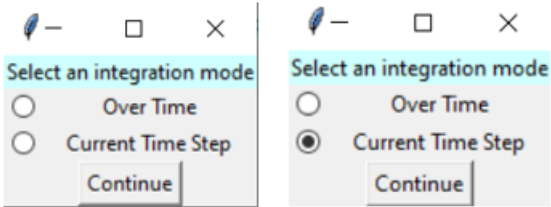
5.6 Integration

TEDs supports two integration modes – “Over Time”, which integrates the plotted variable over a specified space interval at all time steps and generates a plot of how the integrated variable evolves over time, and “Current Time Step”, which integrates over the space interval at only the currently displayed time step. Because time is not displayed on the horizontal axis in the “Current Time Step” mode, the user is free to assign any variable to this axis. Combined with the Batch Initial Condition Tool, the “Current Time Step” mode is useful for sensitivity analyses over single parameters.

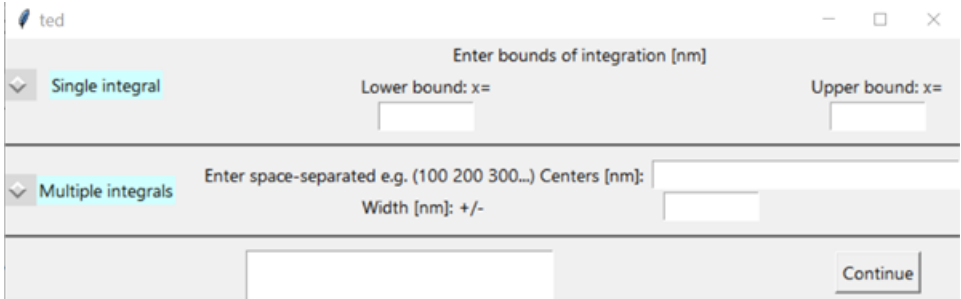
For example, we may want to examine how the carrier density near the front of a Nanowire is affected by the negative charge carrier mobility μ_N , but a plot of the spatial distributions can be quite chaotic.



An integral may summarize the differences more clearly. Upon clicking “Integrate”, the following popup appears for selecting the integration mode.



When “Continue” is clicked, the next popup appears for specifying the lower and upper space boundaries of the integration.



The “Single integral” option performs one integral per dataset over the specified bounds. The “Multiple integrals” performs a set of integrals per dataset, each at a specified centerpoint with given width. These integrals will be cut off at the boundaries if the centerpoint and width would normally take the integral past the boundaries. A system with length 10 000 nm, for instance, will stop integration at 10 000 nm regardless of inputs.

The screenshot shows a software window titled 'ted'. It has two main sections for integration settings. The first section, 'Single integral', is currently disabled. The second section, 'Multiple integrals', is active. It contains two input fields: 'Enter space-separated e.g. (100 200 300...) Centers [nm]:' with the text '0 1000 2000' and 'Width [nm]: +/-' with the text '500'. A 'Continue' button is located at the bottom right of the window.

Figure 34: In this example, integrals over $[0,500]$, $[500,1500]$, and $[1500,2500]$ nm will be performed.

For systems with the “symmetric system” flag active during a simulation, the integration can pass into the symmetric “other half” – for example, Fig. 5.6 will instead span over $[-500, 500]$, $[500,1500]$, and $[1500,2500]$.

For simplicity we will examine a single integral by modifying the inputs from the example above. Note that the "Multiple Integrals" feature allows for single integrals too.

The screenshot shows the same 'ted' software window. In this instance, the 'Single integral' option is selected, and the 'Multiple integrals' option is disabled. The 'Enter space-separated e.g. (100 200 300...) Centers [nm]:' field now contains the value '1000', and the 'Width [nm]: +/-' field still contains '500'. The 'Continue' button remains at the bottom right.

Figure 35: In this example, one integral over $[500,1500]$ nm will be performed.

If the “Current Time Step” mode is selected, a third popup will appear for selecting the variable to be plotted on the horizontal axis.

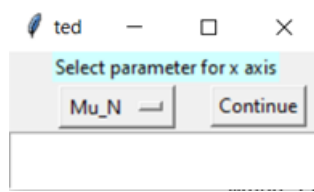
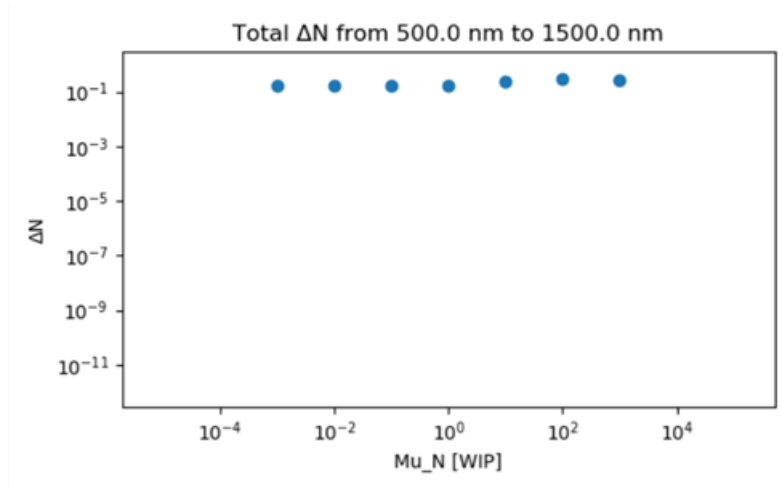


Figure 36: The module parameters determine the available axes - in this case μ_N is selected because the data originate from MSFs with differing mobility values.

The following plot is generated, but adjusting the vertical axis using the “Axis Settings” should prove helpful.



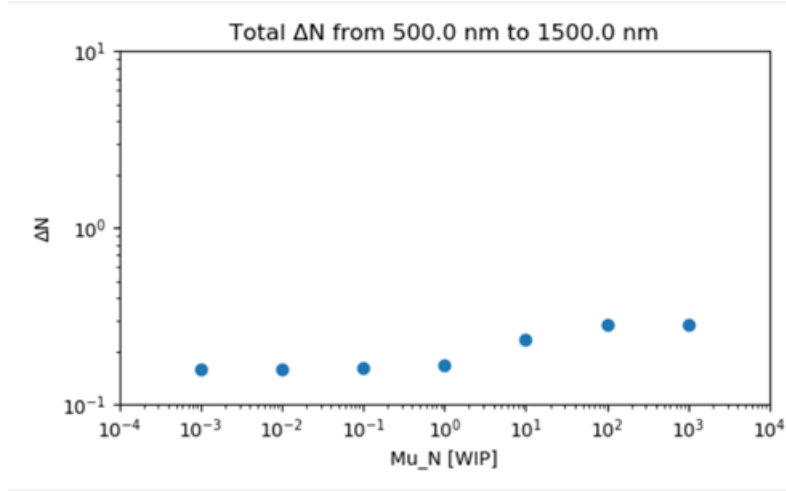


Figure 37: Integration results before and after adjusting using Axis Settings. As expected, high mobility allows negative charge carriers to reach this region of the Nanowire faster, but this effect is only visible when mobility is sufficient to put the diffusion timescale near 2 ns.

5.7 Time Series - Extracting Additional Information from Integrated Data

When certain integrals are computed, TEDs will use the freshly integrated data to make additional calculations. TEDs calls these "Time Series" and displays the results in pop-up windows after the integration is complete.

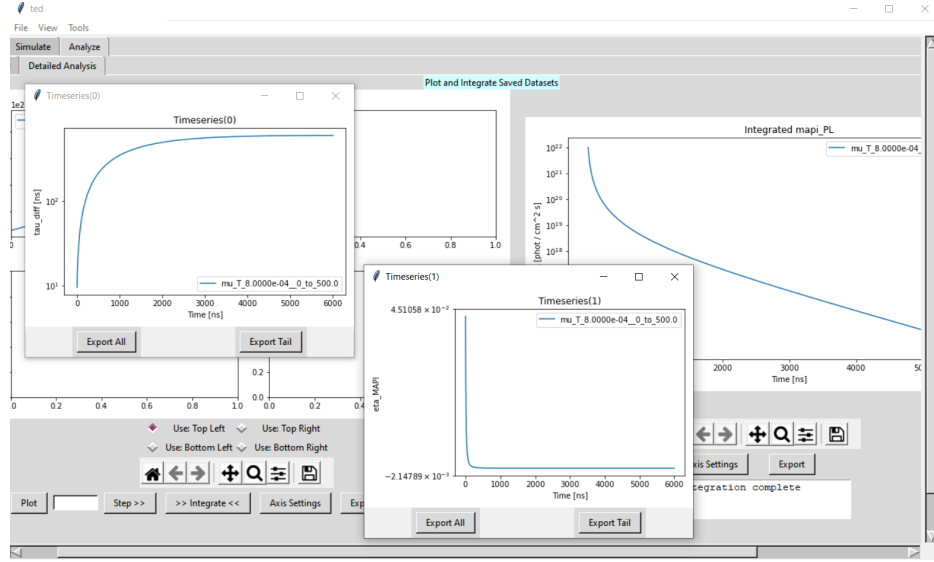
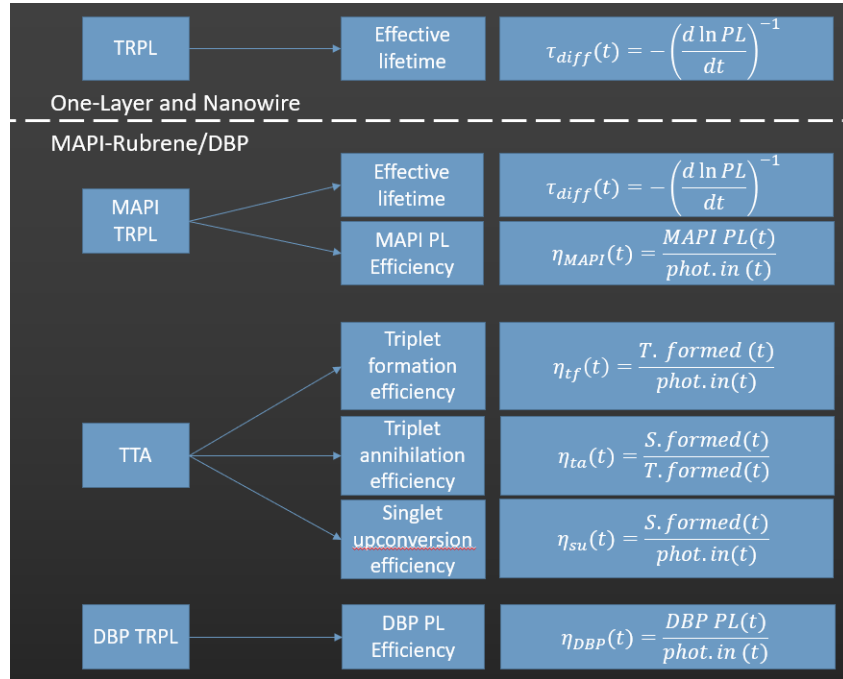


Figure 38: For example, when the TRPL over the MAPI layer of a MAPI-Rubrene/DBP dataset is computed, TEDs also computes the effective carrier lifetime and PL efficiency.

The following table summarizes the Time Series available for each module. Integrate the quantity in the left column to obtain the values in the right column.



6 Adding a Custom Module - Python Familiarity Recommended

6.1 The Module Object Hierarchy

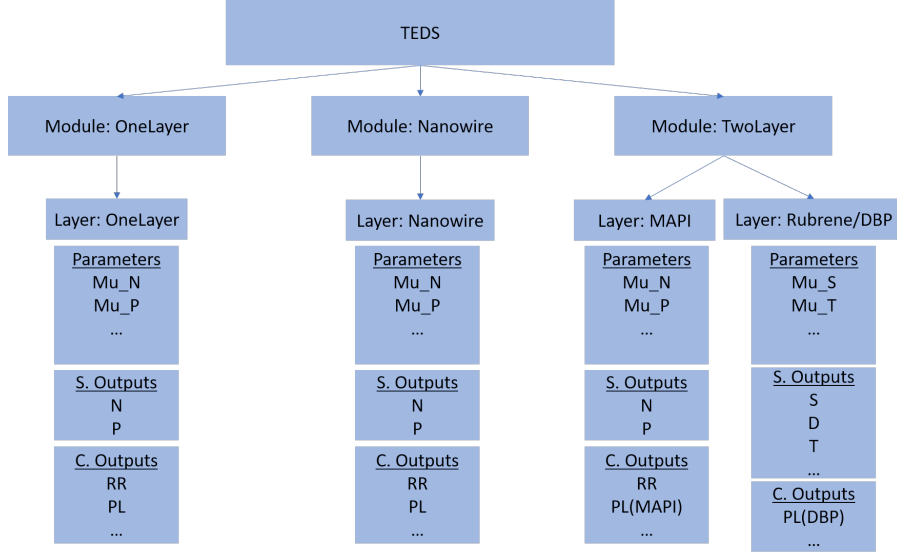


Figure 39: Multilayer structure of TEDs modules.

The *Layer* is the fundamental building block of a TEDs module, representing a self-contained physical system with defined parameters, inputs, and outputs. Parameters and outputs are themselves objects which track internal information such as their names, values, and whether they exist on node edges or centers. Layers manage these objects and may interact with other layers according to the functions packaged in the module.

6.2 Filling Out the Module Template

By extending the `OneD_Model` template class in the file `_OneD_Model.py`, custom modules can be created and simulated with TEDs. A complete module.py file must implement the following functions with the exact arguments specified in the template:

- `__init__()`
- `calc_inits()`

- `simulate()`
- `get_overview_analysis()`
- `prep_dataset()`
- `get_timeseries()`
- `get_IC_carry()`

Additionally, any functions used by a module for calculating values should be defined within that module file. It is highly recommended that these employ Numpy vectorization or similar for compatibility with both 1D and 2D inputs.

Once complete, the `module.py` file must be placed into the `Modules` directory and imported into `main.py`. An entry must also be added into `mod_list()` in `main.py`. The new module can then be selected when TEDs is started.

```
## ADD MODULES HERE
from Modules.Nanowire import Nanowire, tau_diff
from Modules.HeatPlate import HeatPlate
from Modules.Std_SingleLayer import Std_SingleLayer

## AND HERE
def mod_list():
    """
    Tells TEDs what modules are available.

    Returns
    -----
    dict
        {"Display name of module": OneD_Model derived module class}.

    """

    return {"Standard One-Layer": Std_SingleLayer,
            "Nanowire": Nanowire,
            "Neumann Bound Heatplate": HeatPlate}

np.seterr(divide='raise', over='warn', under='warn', invalid='raise')

class Notebook:
    def __init__(self, title):
        """ Create main tkinter object and select module. """
```

The One-Layer module is the recommended starting point for semiconductor materials, while other modules demonstrate how more advanced behaviors may be incorporated.

6.2.1 `__init__()`

This function sets up all informational variables associated with the module and informs TEDs of what parameters or outputs need to be tracked for each layer. For the overall **Module**, the following attributes must be implemented:

- `system_ID`

This is a string specifying the unique identifier of the module. Any string is acceptable, but it is recommended that the `system_ID` reflect the name of the problem the module is intended to represent. For example, the standard One-Layer's `system_ID` is "OneLayer".

- `time_unit`

This is a string specifying the time unit (e.g. "ns") the time grid is calculated in. This field is for plotting display purposes only (i.e. not explicitly used in calculations) but should match the internal time unit used by the module. For example, the time grid actually operates in units of ns for the three provided modules. User-created modules which describe slower physical phenomena may wish to design their solvers on the basis of larger time units.

- `flags_dict`

This dictionary can be used to define custom system flags, whether these values should be toggleable, and their default values. Examples of these are the "Steady State Input" and "Ignore Photon Recycle" flags described in the Initial Tab Overview section.

- `layers`

This dictionary lists the Layer objects that a module is composed of. As discussed in the upcoming section, Layers control their own lists of parameters and outputs.

6.2.2 Defining Layers

Layers are initialized with the following values:

- `length_unit`

This is a string specifying the length unit (e.g. "nm") the space grid for this layer is calculated in. Like the Module `time_unit`, This field is for plotting display purposes only (i.e. not explicitly used in calculations) but should match the internal length unit used by the layer. For example, the space grid operates in units of nm for all layers in the three provided modules. Since the internal time unit is ns, this means all incoming parameters for these modules must be converted to the nm-ns unit system and all raw outputs will come out with these units. See the `convert_in` and `iconvert_in` sections for additional details.

- `params`

This is a dictionary of Parameter objects corresponding to the parameters which exist in this layer. For example, in the MAPI-Rubrene/DBP module, the MAPI Layer contains parameter objects for N and P mobilities while the Rubrene Layer contains similar objects for T, S, and D mobilities. Both material properties, such as mobilities, and initial conditions, such as $\Delta N(t=0)$, should be defined here.

- `simulation_outputs`

This dictionary specifies outputs which are returned directly when TEDs simulates a system created from the module and uses `Output()` objects to store information regarding these outputs. For example, the One-Layer module simulates carrier densities; therefore the simulation outputs are $N(x,t)$ and $P(x,t)$. The display information is used to populate plot fields.

Any output here should have a corresponding Parameter from which initial values may be calculated. For example, the Output N has Parameters $\Delta N(t=0)$ and N_0 which together calculate $N(t=0)$.

- `calculated_outputs`

This dictionary specifies outputs which are calculated from those listed in `simulation_outputs` but not simulated directly. For example, TRPL is not calculated during a simulation, but rather from ΔN and ΔP after the fact.

`simulation_` and `calculated_outputs` are also combined into a single outputs dictionary for convenience.

- `convert_in`

This dictionary specifies conversion factors that should be used to change from the units entered into the interface to the internal solver units. Generally, this dictionary should be used to correct mismatches between common length and time units (such as inputting cm^{-3} for carrier densities) and the length and time scale used by the module (i.e. the space grid is in nm and works with N in units of nm^{-3} rather than their cm counterparts). One conversion factor must be defined for each Parameter and Output. Quantities which do not need conversions should be assigned a conversion factor of 1.

- `iconvert_in`

This dictionary specifies conversion factors that should be applied in addition to `convert_in` when integrals are calculated. For example, N needs a `convert_in` entry to switch between cm^{-3} and nm^{-3} , but when integrated over length, N switches between cm^{-2} and nm^{-2} instead. The `iconvert_in` entry addresses the difference between $length^{-2}$ to $length^{-3}$. One conversion factor must be defined for each Output. Quantities which do not need conversions should be assigned a conversion factor of 1.

6.2.3 Defining Parameters

Parameters are initialized with the following values:

- `units`

This is a string specifying the units to be displayed with this parameter. Use this field to indicate the units this parameter should be entered and the conversion dictionaries to convert into internal solver units.

- `is_edge`

Whether this parameter should be assigned to space node centers or space node edges.

- `valid_range`

A tuple `[a,b]` used to restrict the allowed values for this parameter. Inputting values smaller than `a` or larger than `b` will raise an error. For example, only positive (`a=0`, `b=infinity`) mobility values are allowed. This is useful for screening out nonsense inputs.

- `is_space_dependent`

Setting this flag to False will exclude this parameter from all input methods except for the Fast Parameter Entry Tool, effectively allowing only single values to be assigned to this parameter. Surface parameters, such as the front and back surface recombination velocities, should have this flag deactivated.

6.3 `calc_inits()`

This function is used by TEDs to obtain all initial conditions needed by the solver and must return a dictionary {"param name":np.ndarray} containing one 1D array with the initial value of each space node per output in `simulation_outputs`. Use of the module's parameter dictionaries to access material parameter values, `convert_in` for unit conversions, and `grid_x_nodes`, and/or `grid_x_edges` to access the space grids is highly recommended.

6.4 `simulate()`

This function is used by TEDs to call the function(s) responsible for (1) preprocessing inputs, (2) simulating the data, and (3) writing the data to output files. It is not required to use all of `simulate()`'s arguments, and any implementation which completes the three mentioned items is acceptable. For the One-Layer, all three of these tasks are completed by the driver function `ode_nanowire()`.

6.5 `get_overview_analysis()`

Like `calc_inits()`, this function must return a dictionary {"param name":np.ndarray} containing one array per output in `simulation_` and `calculated_outputs`. The `u_read()` helper function is recommended for reading in a section of data over a specified time and space range.

6.6 `prep_dataset()`

This function is used to do preprocessing of data prior to plotting and must return a 1D array of data (a single time step) to be plotted or a 2D array (time and space) to be integrated. For instance, in some modules the TRPL requires multiplying the radiative recombination by a weighting function to account for photon recycling prior to plotting. For most outputs preprocessing is simply reading the appropriate entry of `sim_data` or calling an appropriate calculation function.

6.7 get_timeseries()

This function is used to calculate the time series associated with specific outputs. A list of tuples should be returned - one tuple per time series. Each tuple contains two values - first value the name of the time series and second value a 1D np.ndarray containing the time series values.

6.8 get_IC_carry()

This function is used to assign current data values to param_dict for regenerating MSF files using the carryover feature. This function must write into the appropriate entry of param_dict using include_flags to determine how to respond to which items are selected in the carryover feature.