Transient Electron Dynamics Simulator

User Manual

v 1.1.

Contents

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 SciPy odeint() methods, themselves based on the LSODA methods of the ODEPACK FORTRAN77 library, to evolve initial carrier distributions forward in time, the systems of differential equations can be solved numerically with a relatively simple Python solver.

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 singular 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 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:

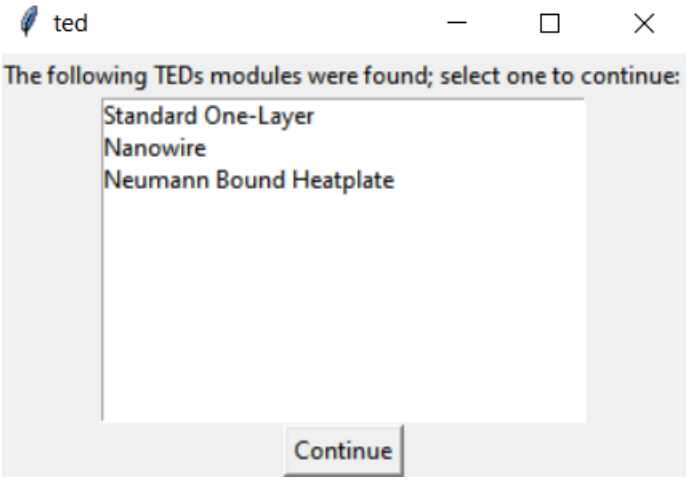


Figure 1: Three modules are available by default: a general-purpose one-layer material, a nanowire with axial rather than surface emission, and a "toy" one-dimensional heat transfer problem.

TEDs scans the Modules directory for available module files and lists them in the Module Selection window. After selecting a module, TEDs performs some verifications 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:

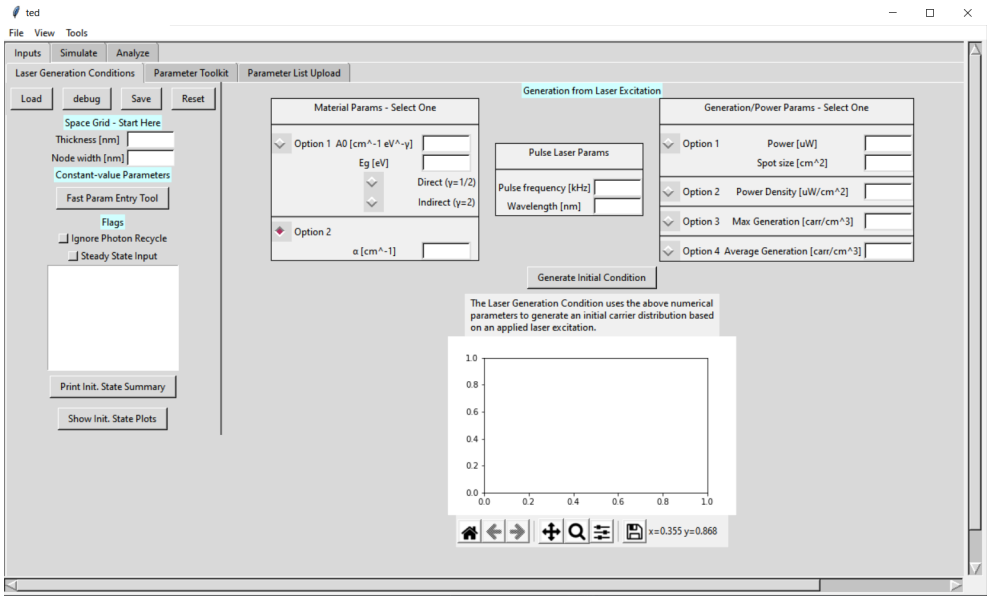
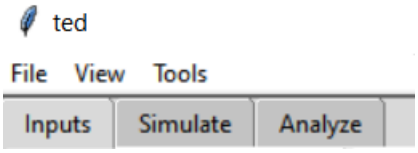


Figure 2: Main interface.

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.



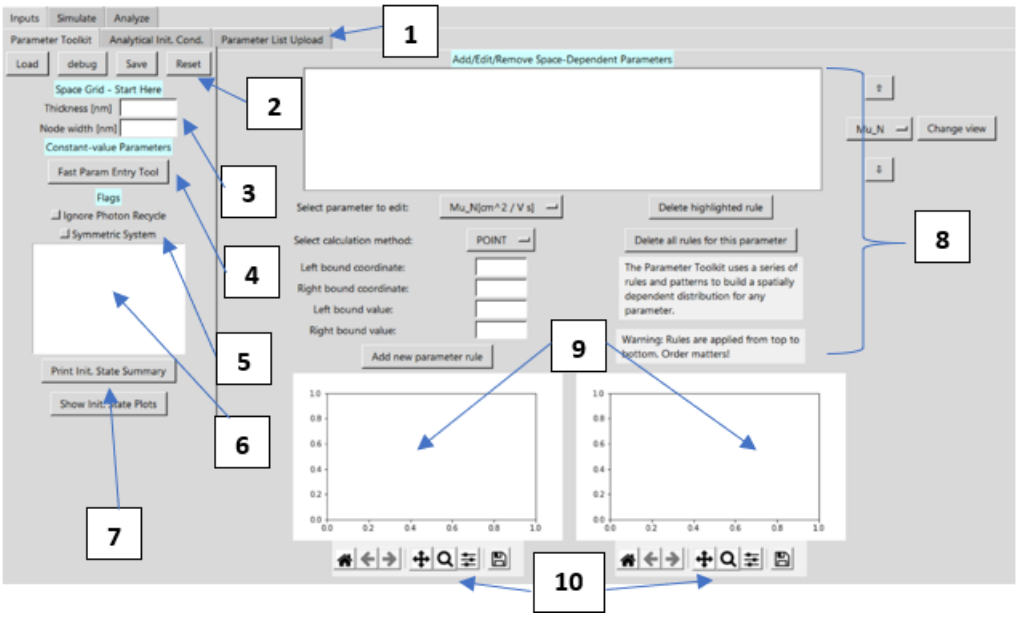
The top left corner also contains three tabs — Inputs, Simulate, and Analyze, which correspond to the three main simulation stages of preparing an initial condition and parameter set, simulating the evolution of the initial condition over time, and viewing and analyzing the results. Each of the subinterfaces 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 like any other. 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 width 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:



1. Initial Condition Subtabs

   The Inputs Tab itself contains three subtabs – Laser Generation Condition (formerly Analytical Initial 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 [8].

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.

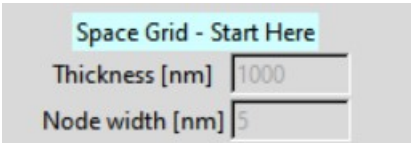


Figure 3: A "locked in" space grid of length 1000 nm and node width 5 nm. This grid consists of 200 nodes whose centers are at 2.5, 7.5, 12.5...997.5 nm and whose edges are at 0, 5, 10...1000 nm.

   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.

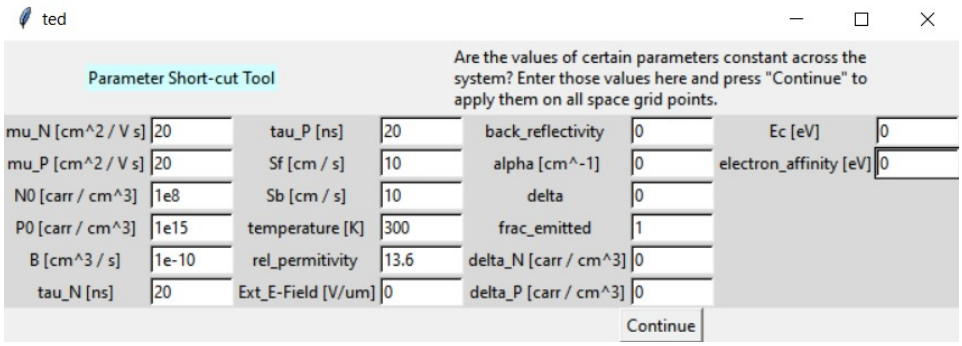


Figure 4: Example list of values for each of the standard one-layer model's parameters. When "Continue" is clicked, all nodes and edges are assigned these values.

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

Module-specific flags can be toggled here. By default, two flags are available to the one-layer and nanowire modules.

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

The "Steady State Input" flag alters how the initial carrier densities ΔN and ΔP are treated. When active, TEDs assumes that continuously, 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 density rate rather than an initial carrier density distribution (as from a single laser pulse)

Furthermore, both the one-layer and nanowire modules define a hidden "symmetric system" flag – always active for the nanowire and inactive

for the one-layer as according to their experimental setups (i.e. the nanowire is excited at its center whereas the one-layer is excited at its surface). Any module 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. 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.*

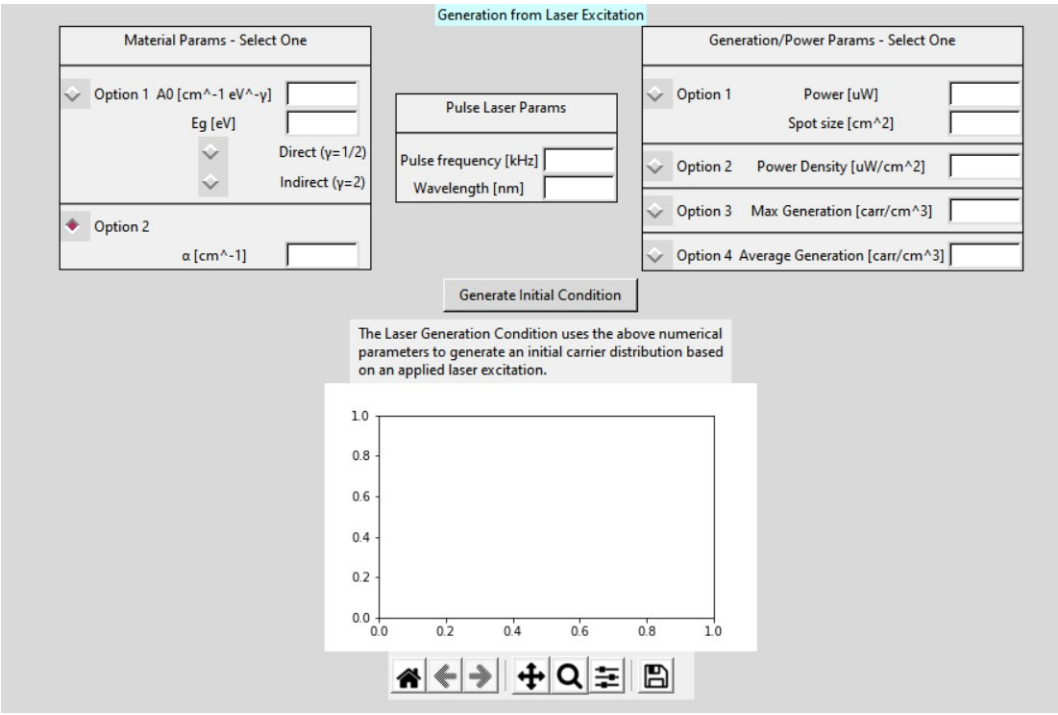
9. Plots

   Each of these two plots present a visualization of certain parameters of the initial state – a user-selected "snapshot" in the left plot, and the most recently changed parameter in the right plot.

10. Plot Options

    These toolbars offer resizing and scaling of the Initial State Plots [9] as well as options to export images of the plots.

3.2 Laser Generation Conditions (LGC)



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

First, the material parameter α may be calculated from the below or entered directly.

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

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

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

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

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

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

TEDs automatically performs all unit conversions needed to obtain ΔN and ΔP in $\frac{carriers}{nm^3}$ 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:

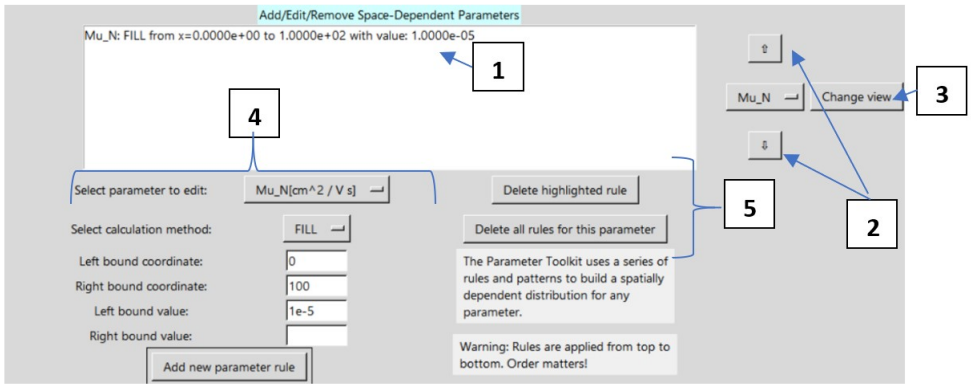


Figure 5: Parameter toolkit with one rule entered for parameter μ_n. This rule, for example, assigns the value $10^{-5}\frac{cm^2}{Vs}$ to all nodes spanning 0 nm to 100 nm.

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.

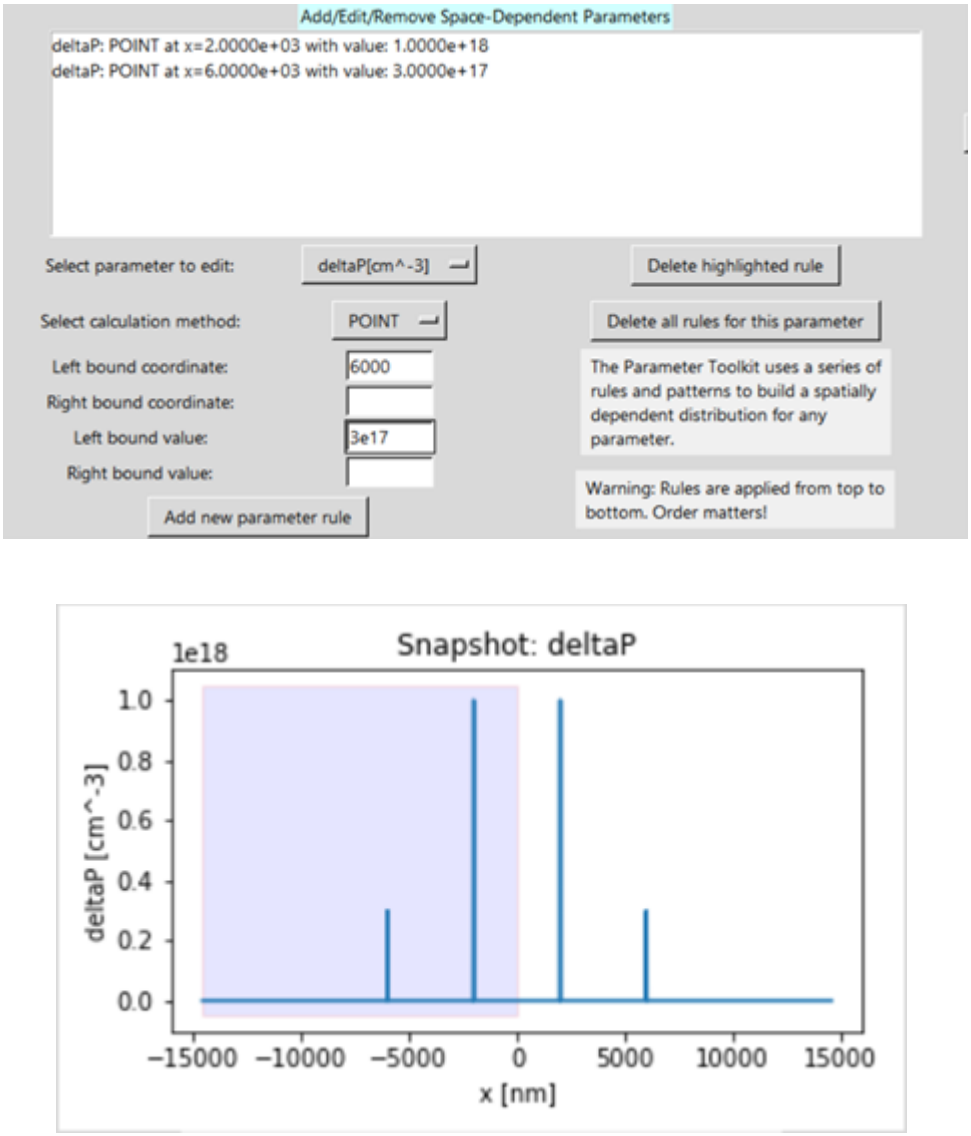
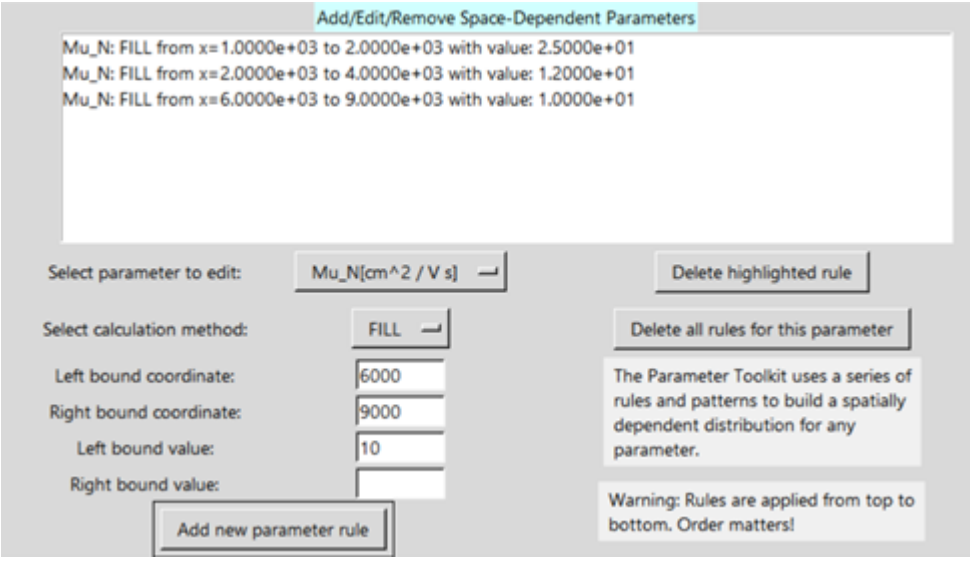
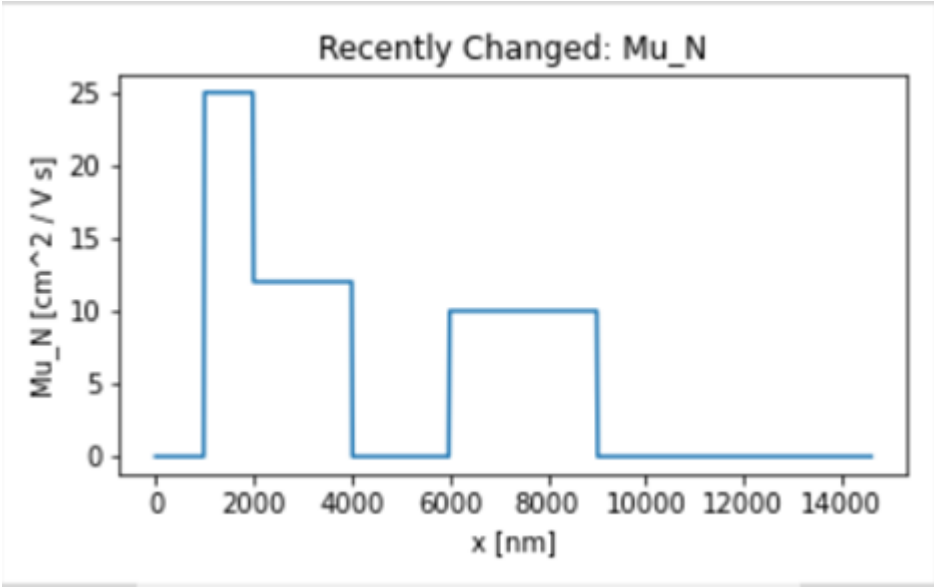


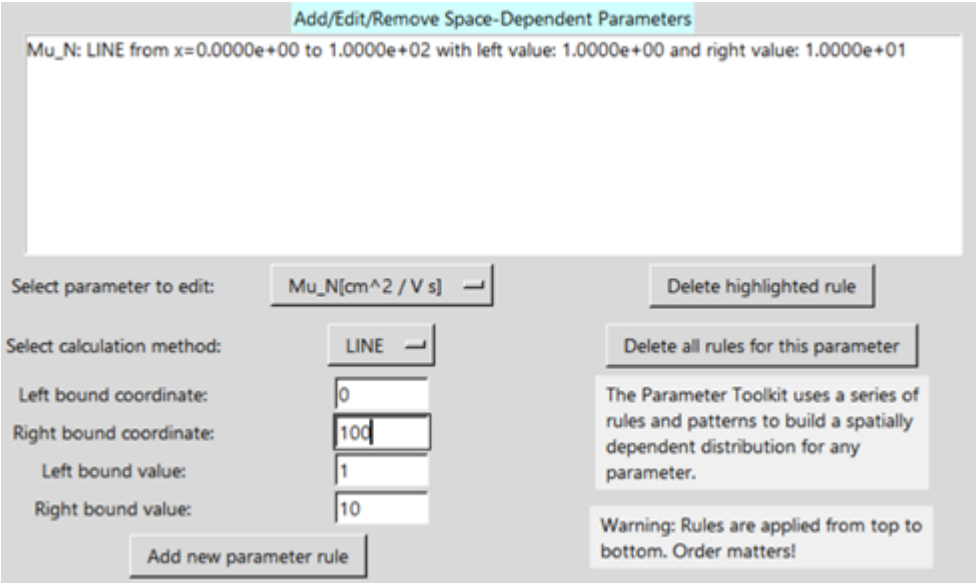
Figure 6: In this example with the nanowire module, the symmetric system flag is active - note the mirrored half of the system represented in the plot.

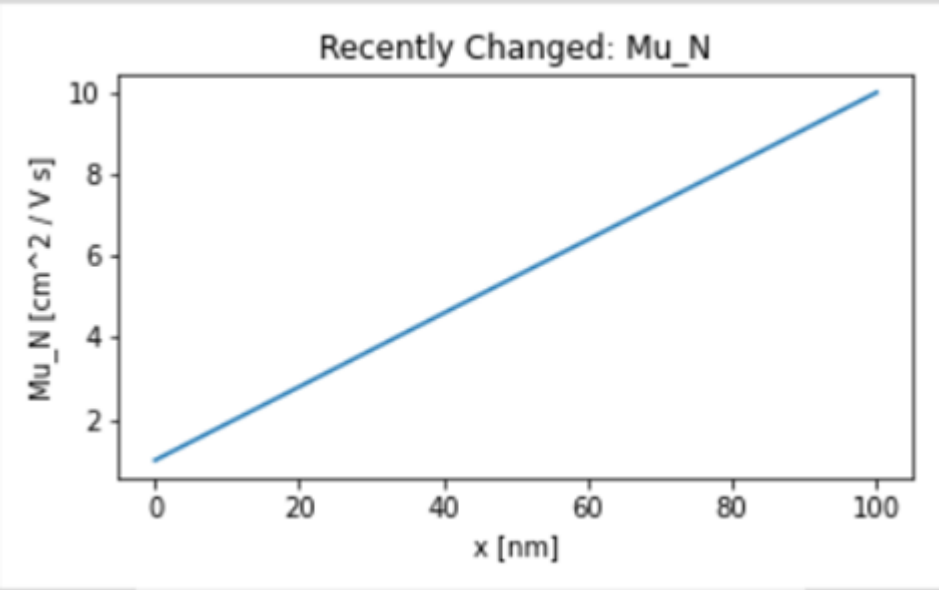
The FILL method fills every space point from the Left Bound Coordinate to the Right Bound Coordinate with the Left Bound Value. This method is useful for specifying regions with constant values and is a special case of the LINE method.



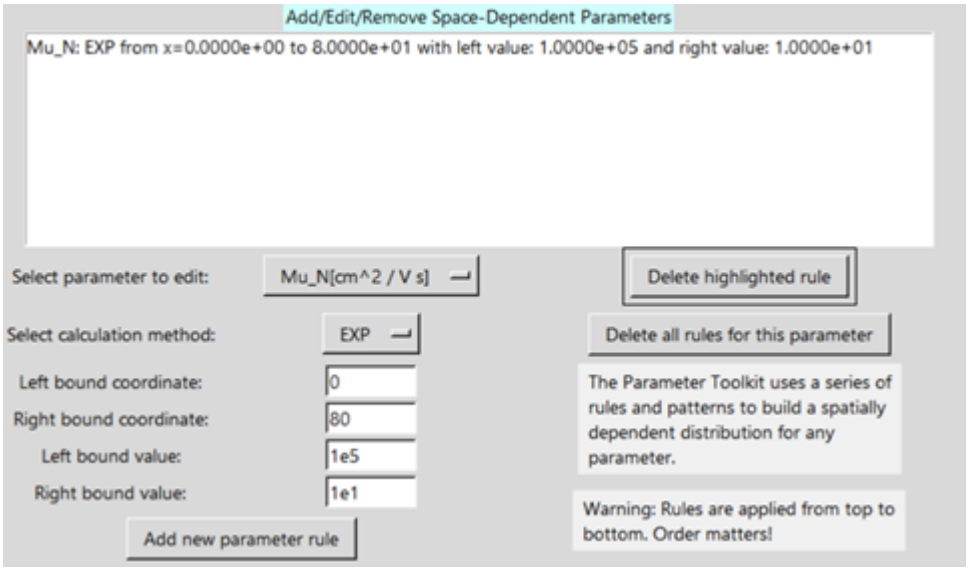


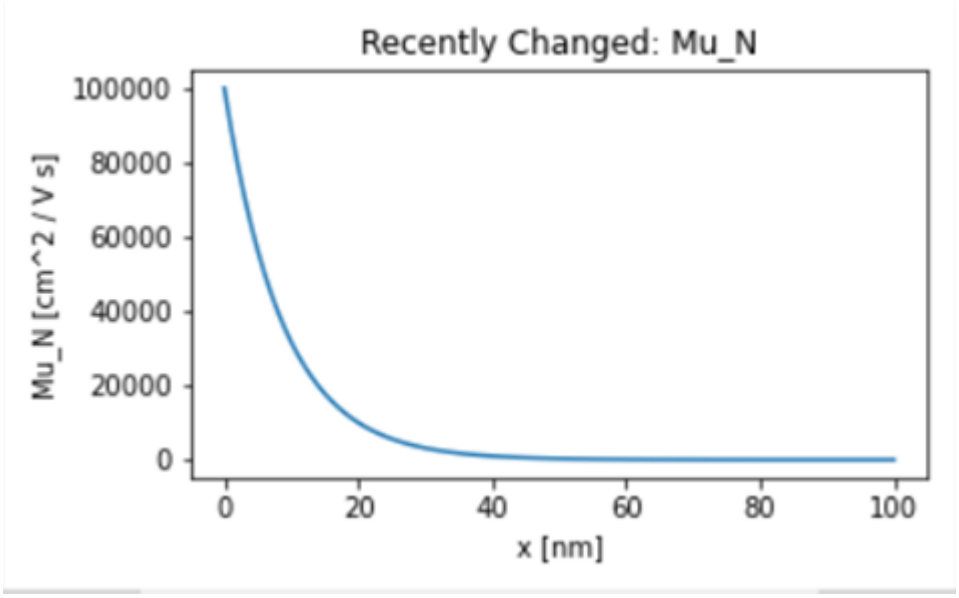
The LINE method assigns the Left Bound Value to the Left Bound Coordinate, assigns the Right Bound Value to the Right Bound Coordinate, and performs linear interpolation to assign values to all intermediate space points. Each intermediate point differs from its neighbors by a common difference value.



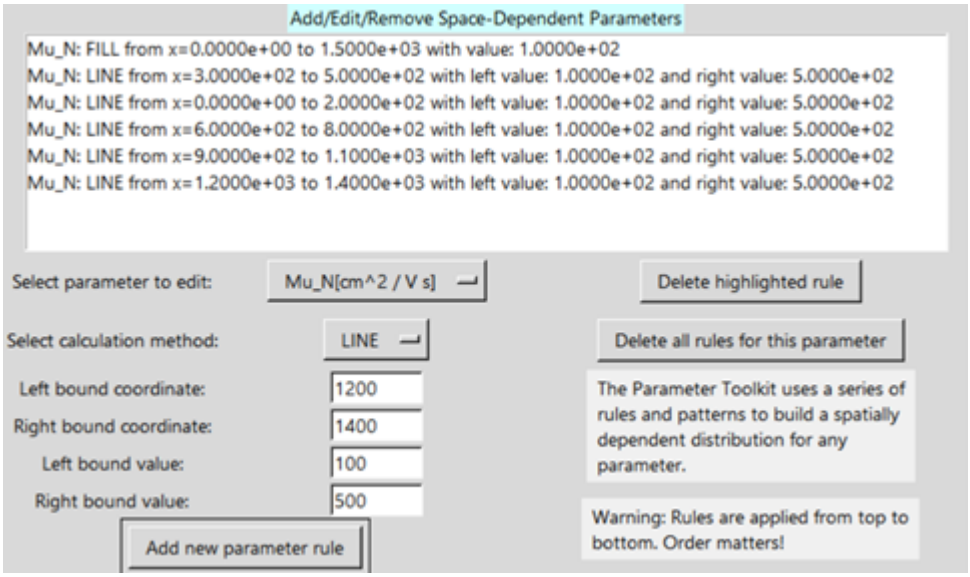


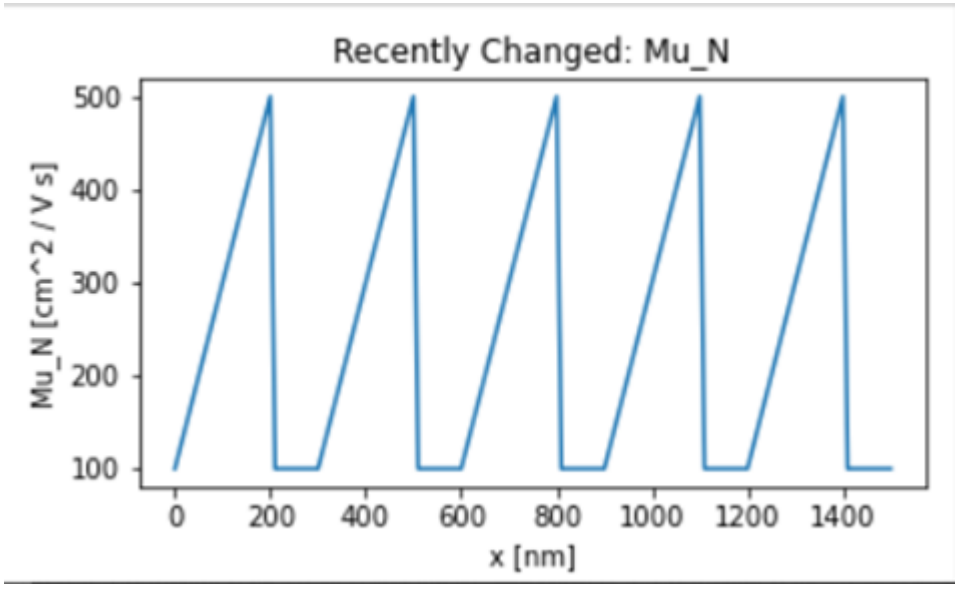
The EXP method is similar to LINE, but intermediate space points are instead filled by exponential interpolation between the left and right bounds. Each intermediate point differs from its neighbors by a common ratio.





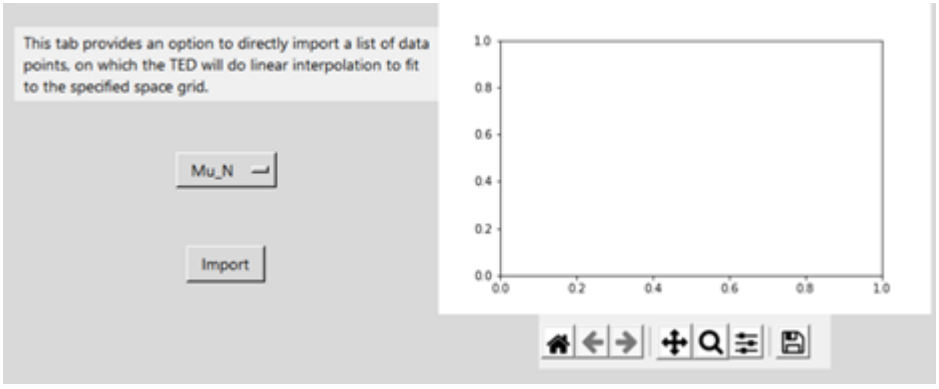
These four methods can be combined to set up complex distributions.


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[cm^2 / V s]
Delete highlighted rule
Select calculation method:
LINE
Delete all rules for this parameter
Left bound coordinate:
1200
Right bound coordinate:
1400
Left bound value:
100
Right bound value:
500
The Parameter Toolkit uses a series of rules and patterns to build a spatially dependent distribution for any parameter.
Add new parameter rule
Warning: Rules are applied from top to bottom. Order matters!


Recently Changed: Mu_N
Mu_N [cm^2 / V s]
500
400
300
200
100
0
200
400
600
800
1000
1200
1400
x [nm]

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



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

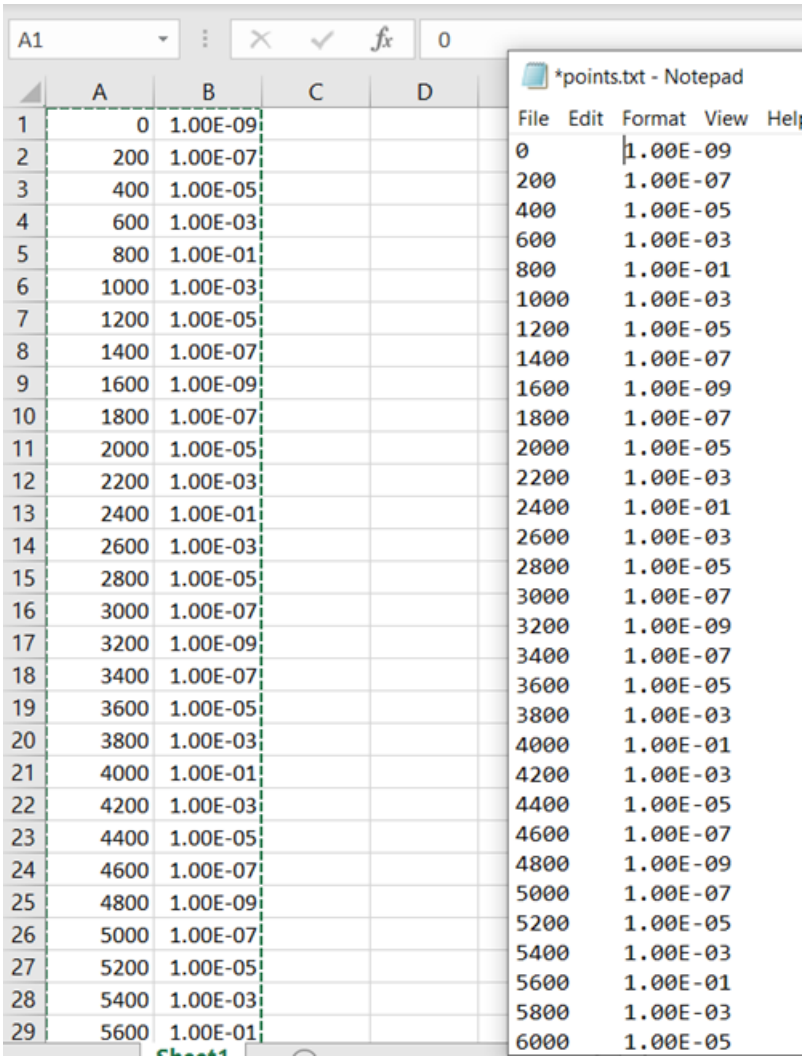
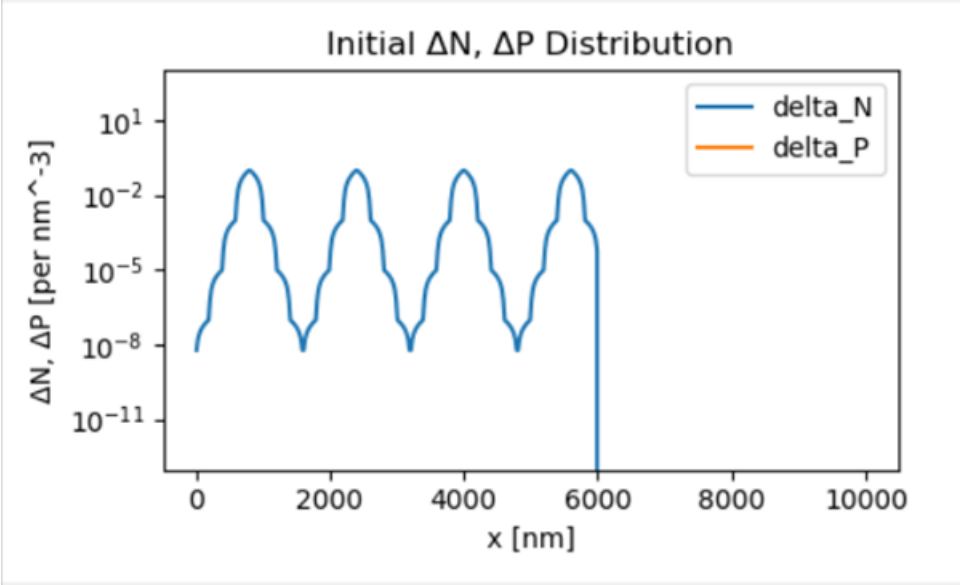


Figure 7: Copying data from a spreadsheet program such as Excel can automatically provide the necessary tab-separated format.



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.

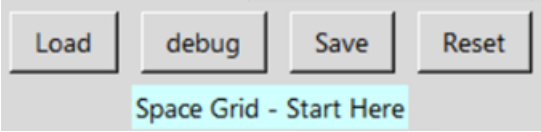


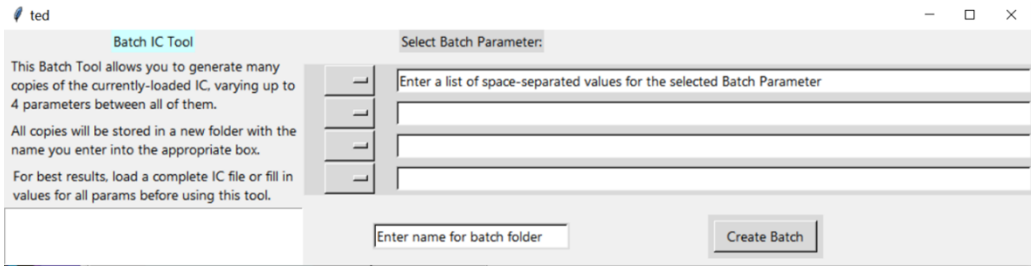
Figure 8: Also included is a debug button - this prints out log messages but serves no other function.

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.

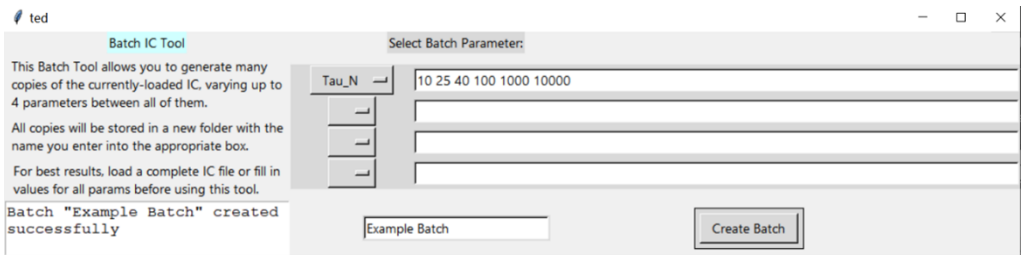
Saving an MSF without all system parameters entered is possible. In that case, all of the distributions would be saved as their default values – zeroes.

3.7 The Batch MSF Tool

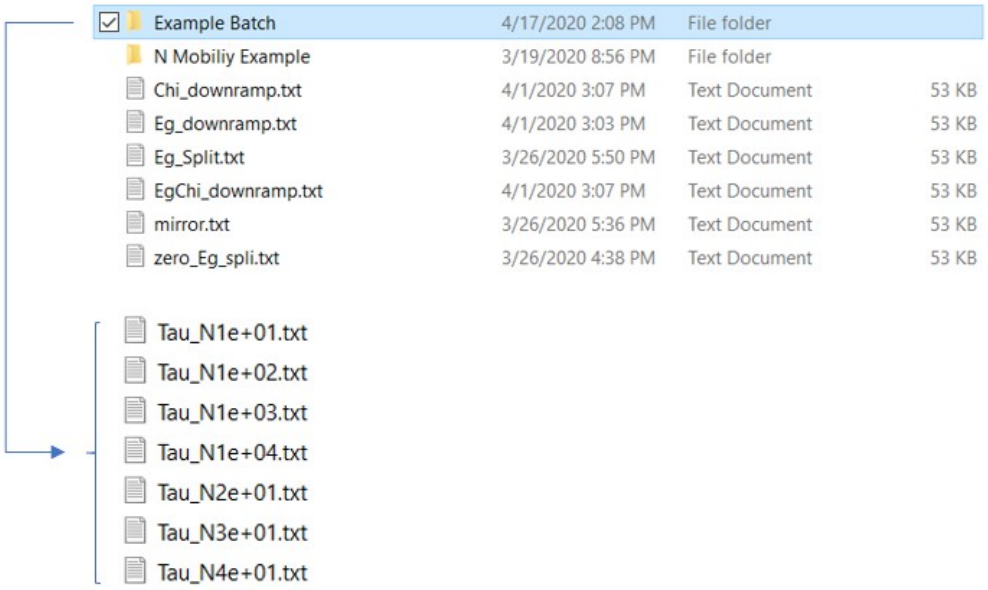
In many situations, such as sensitivity analyses, it is useful to generate many model state files over a rectangular parameter space. In the "Tools" menu, TEDs offers the Batch Model State File Tool, which provides a fast method to generate many such copies based on an existing initial state.



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.



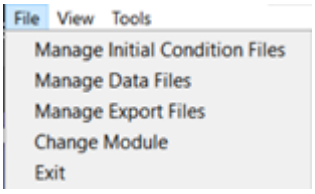
When "Create Batch" is clicked, a directory is created within the Initial Directory, alongside any existing MSFs. Each file in the batch is procedurally named using the values of the varied parameters it has adopted.



Up to four parameters may be varied at a time.

3.8 Removing Files

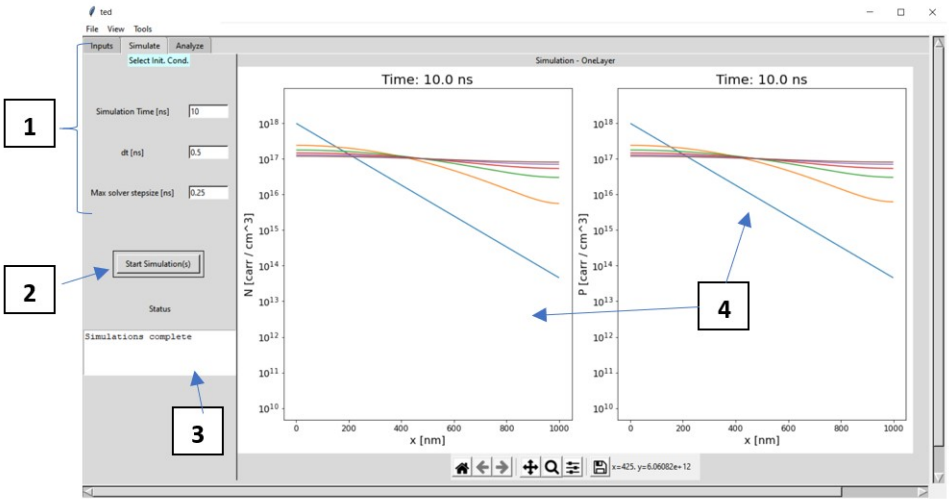
In the top left corner menu under "File", TEDs provides a series of shortcuts to each of the working directories.



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:



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.

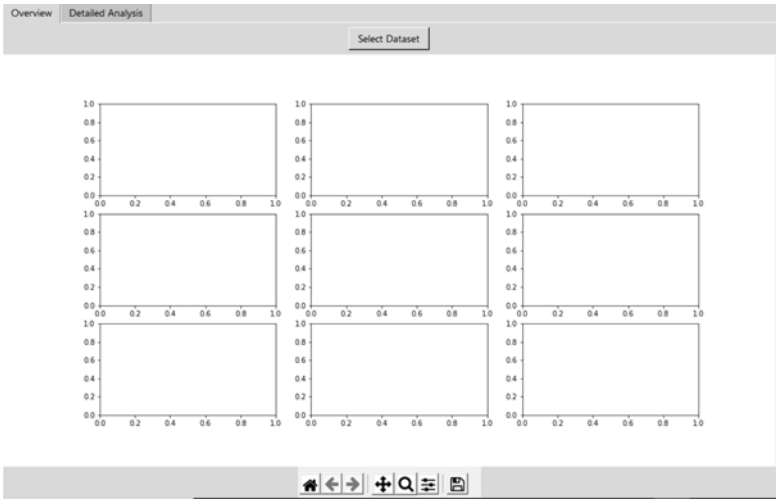
4. Results Windows

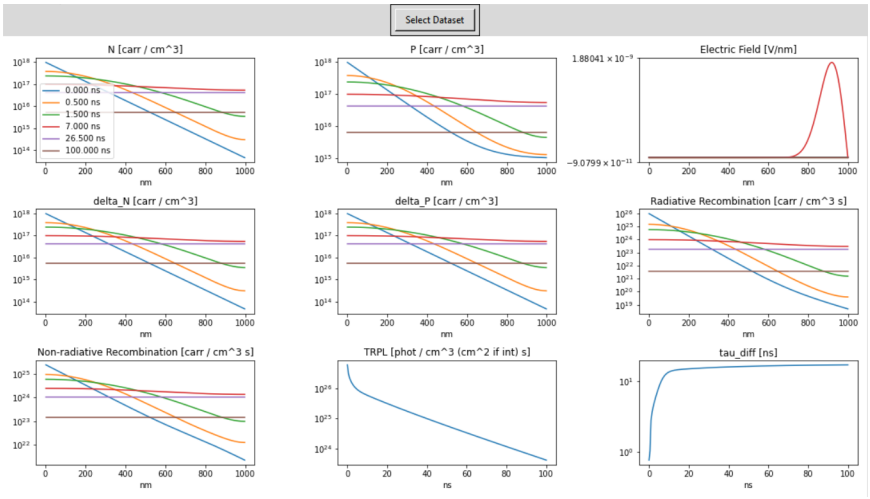
   These plots display snapshots of time steps taken during the simulation, a useful first glance of how the model is evolving over time. As with the Initial tab plots, these plots have a toolbar for basic resizing and exporting.

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}

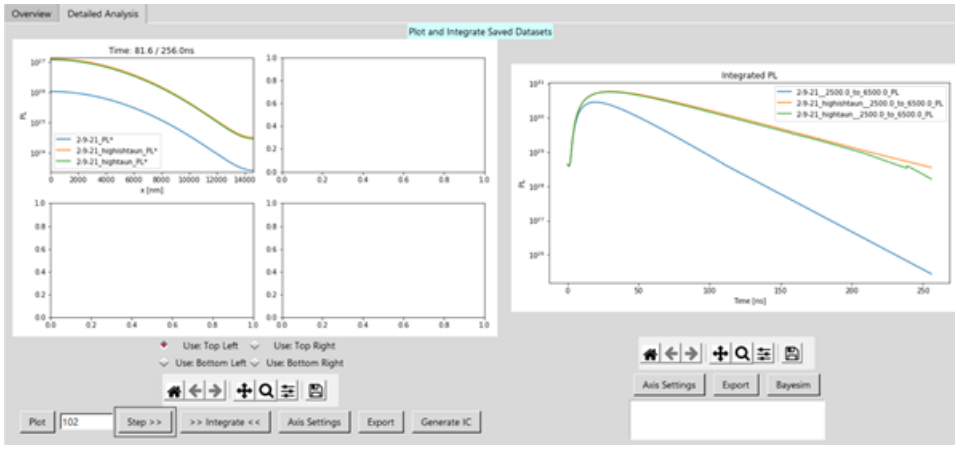


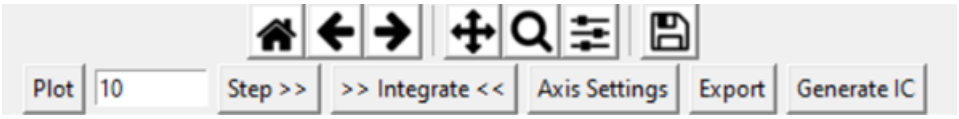


To view other timesteps or TRPL over other ranges, for example, the Detailed Analysis tab should be used.

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.





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.

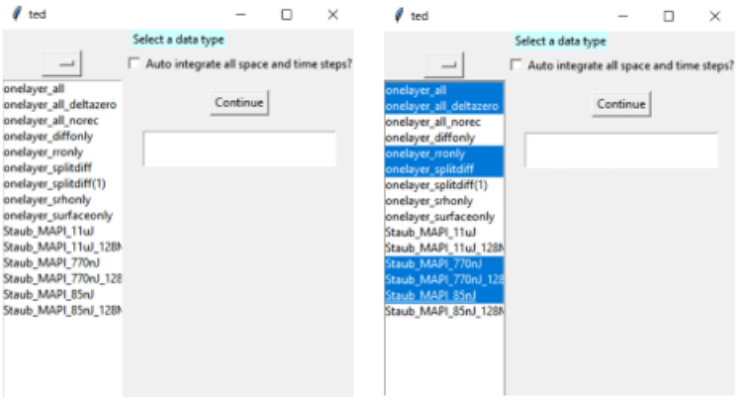
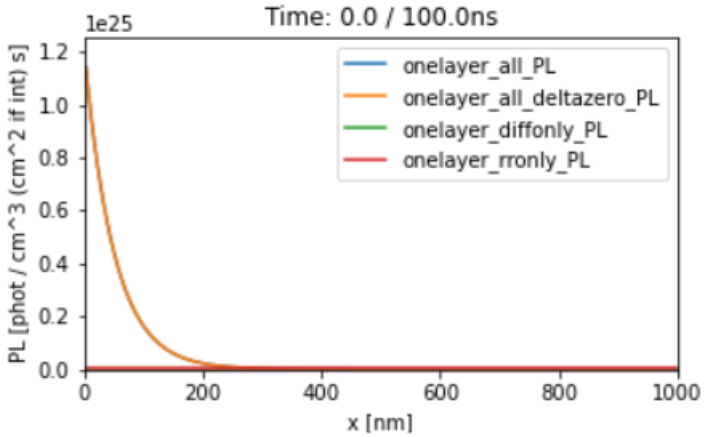


Figure 9: To select a single dataset, click its name in the box. To select multiple datasets, hold the Ctrl key and drag the cursor over every desired dataset.

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.



One major 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.

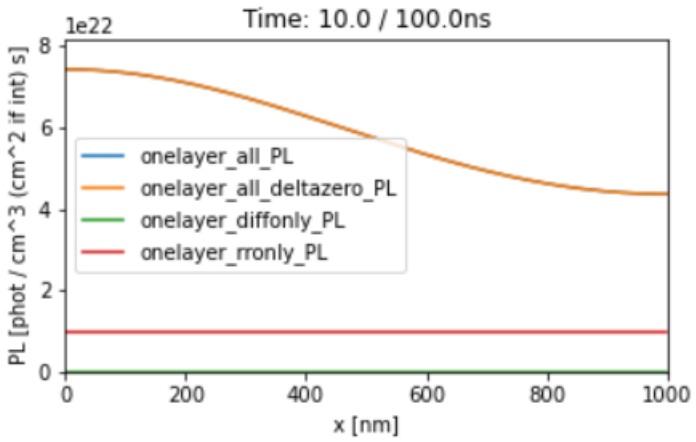
If datasets with different total time or time step size should be compared, plotting each on a different plot is recommended.

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.

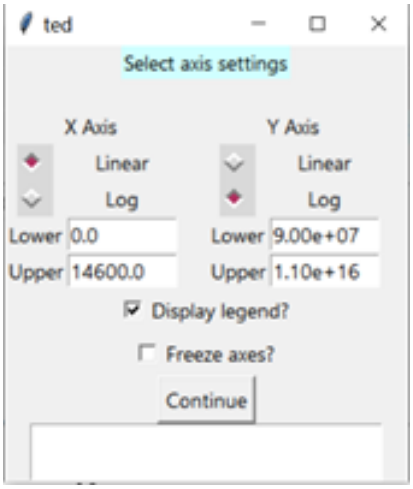
With a time step size of 0.5 ns, for example, entering 20 will advance the datasets by 10.0 ns. each time the "Step" button is clicked.



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.



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.

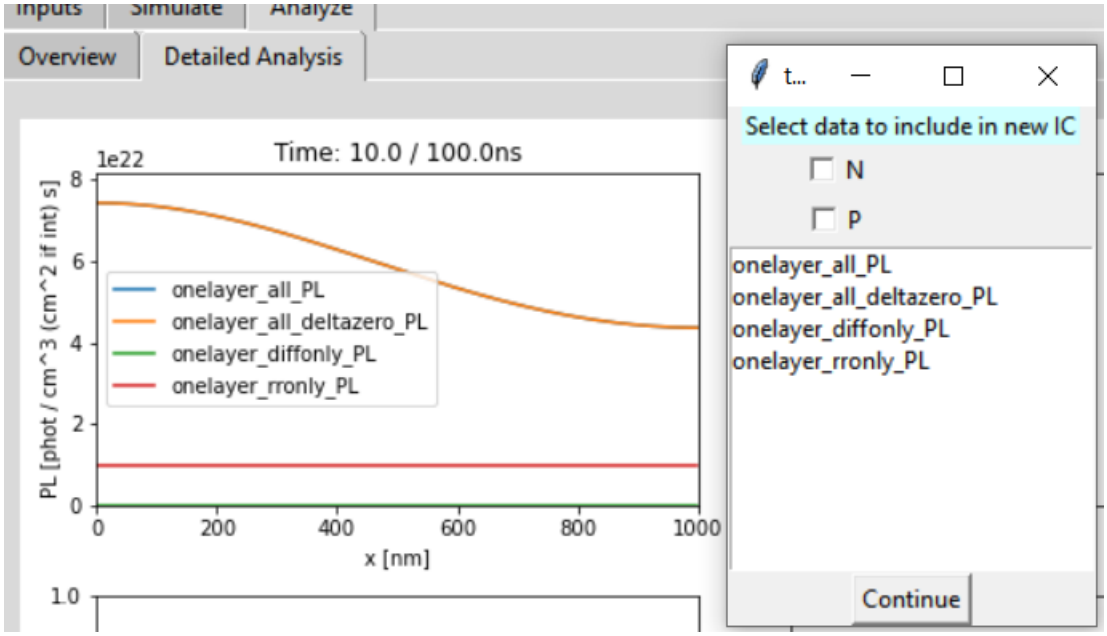
5.4 Exporting Data

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

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.



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.

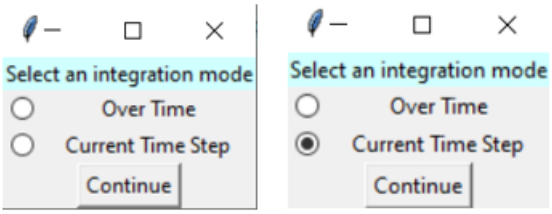
These MSFs can then be modified further using the Initial tab or simulated using TEDs' other tabs.

5.6 Integration

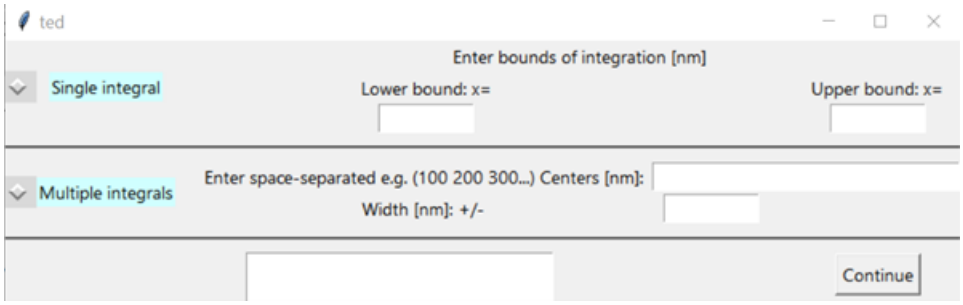
TEDs supports two integration modes – "Over Time", which integrates the plotted variable over a specified space interval at all time steps and generates a plot of how the integrated variable evolves over time, and "Current Time Step", which integrates over the space interval at only the currently displayed 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 μN, but a plot of the spatial distributions can be quite chaotic.

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.

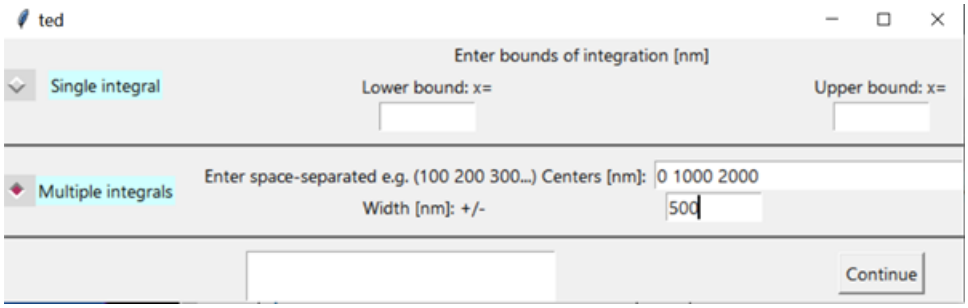
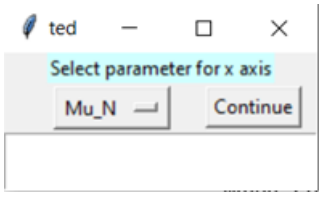


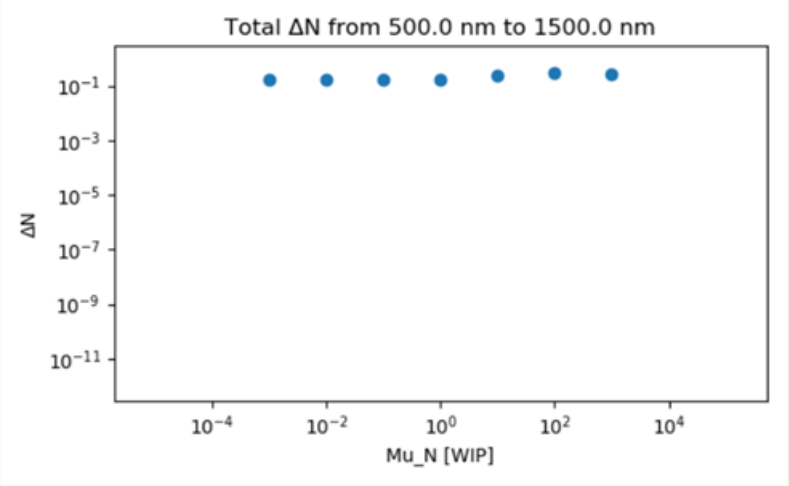
Figure 10: In this example, integrals over [0,500], [500,1500], and [1500,2500] nm will be performed.

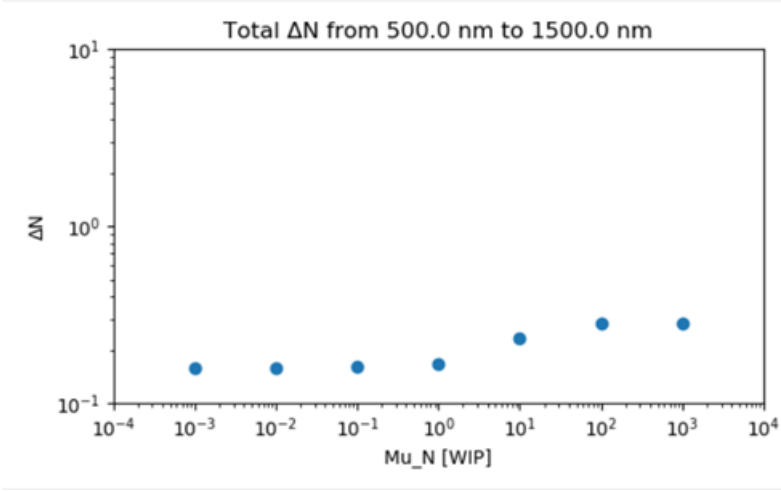
For systems with the "symmetric system" flag active during a simulation, the integration can pass into the symmetric "other half" – for example, Fig. 5.6 will instead span over [-500, 500], [500,1500], and [1500,2500].

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



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


Total ΔN from 500.0 nm to 1500.0 nm
ΔN
10^{-1}
10^{-3}
10^{-5}
10^{-7}
10^{-9}
10^{-11}
10^{-4}
10^{-2}
10^{0}
10^{2}
10^{4}
Mu_N [WIP]


Total ΔN from 500.0 nm to 1500.0 nm
ΔN
10^{1}
10^{0}
10^{-1}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
10^{4}
Mu_N [WIP]

6 Adding a Custom Module

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_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 the HeatPlate module is provided as a proof-of-concept for TEDs's re-usability.

6.1 __init__()

This function sets up all informational variables associated with the module and informs TEDs of what parameters or outputs need to be tracked. 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".

- length_unit and time_unit

These are strings specifying the length unit (e.g. nm, m, km) the space grid is calculated in and the time unit (e.g. ns) the time grid is calculated in. These fields are for display purposes only (i.e. not explicitly used in calculations). However, the space and time grids are assumed to take on the same units as the parameters (e.g. these are in nm and ns for the One-Layer).

- param_dict

  This is a dictionary {"parameter name":Parameter()} that stores an informational Parameter() object for each parameter. The Parameter() object contains information regarding the displayed units of the parameter, whether it should be calculated at a space node center or node edge, and the working values of the parameters.

- flags_dict

  This dictionary {"internal name":"displayed name"} can be used to define custom system flags, their default values, and whether these values should be toggleable. Examples of these are the "Steady State Input" and "Symmetric System" flags described in the Initial Tab Overview section.

- simulation_outputs_dict

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

- calculated_outputs_dict

  This dictionary specifies which outputs are calculated from those listed in simulation_outputs_dict 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_dict are also combined into a outputs_dict for convenience.

- convert_in_dict

This dictionary specifies conversion factors that should be used to change from the values entered into the interface to the actual values used by calculation functions. 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 rather than cm). One conversion factor must be defined for each input in param_dict and each output in simulation_ and calculated_outputs_dict.

6.2 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_dict. Use of the module's param_dict to access material parameter values, convert_in_dict for unit conversions, and grid_x_nodes, and/or grid_x_edges to access the space grid is highly recommended.

6.3 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().

6.4 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_dict. The u_read() helper function is recommended for reading in a section of data over a specified time and space range.

6.5 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, the TRPL requires multiplying the radiative recombination by a weighting function prior to plotting. For most

outputs preprocessing is simply reading the appropriate entry of sim_data or calling the appropriate calculation function.

6.6 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.