Transient Electron Dynamics Simulator (TEDs)

v. 1.0.0

User Manual

Introduction

The Transient Electron Dynamics Simulator (TEDs) was 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. A solver can get the job done, but a solver operating underneath the hood of a program 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. TEDs offers an interface for the initial state-simulation-data analysis workflow and supports a variety of time-resolved problems.

Table of Contents

[Getting Started 3](#_Toc37964699)

[Setting up TEDs Directories/Folders 3](#_Toc37964700)

[Familiarizing with the Layout 4](#_Toc37964701)

[The Inputs Tab - Creating Initial Condition Files 5](#_Toc37964702)

[Overview 5](#_Toc37964703)

[Analytical Initial Conditions (AIC) 6](#_Toc37964704)

[Heuristic Initial Conditions (HIC) 7](#_Toc37964705)

[Heuristic Mathematical Methods 8](#_Toc37964706)

[Explicitly Defined Initial Conditions (EIC) 13](#_Toc37964707)

[Combining Initial Condition Methods 14](#_Toc37964708)

[Saving and Loading Initial Condition Files 15](#_Toc37964709)

[The Batch Initial Condition Tool 16](#_Toc37964710)

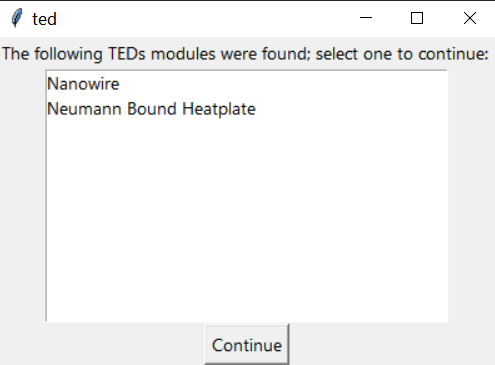
[Running Simulations 19](#_Toc37964711)

[Analyzing and Integrating Simulation Data 21](#_Toc37964712)

# Getting Started

## Choosing a Simulation Model

Although TEDs is originally designed around the one-dimensional nanowire problem, a variety of other modules is also supported. When TEDs is launched, a Module Selection window appears:

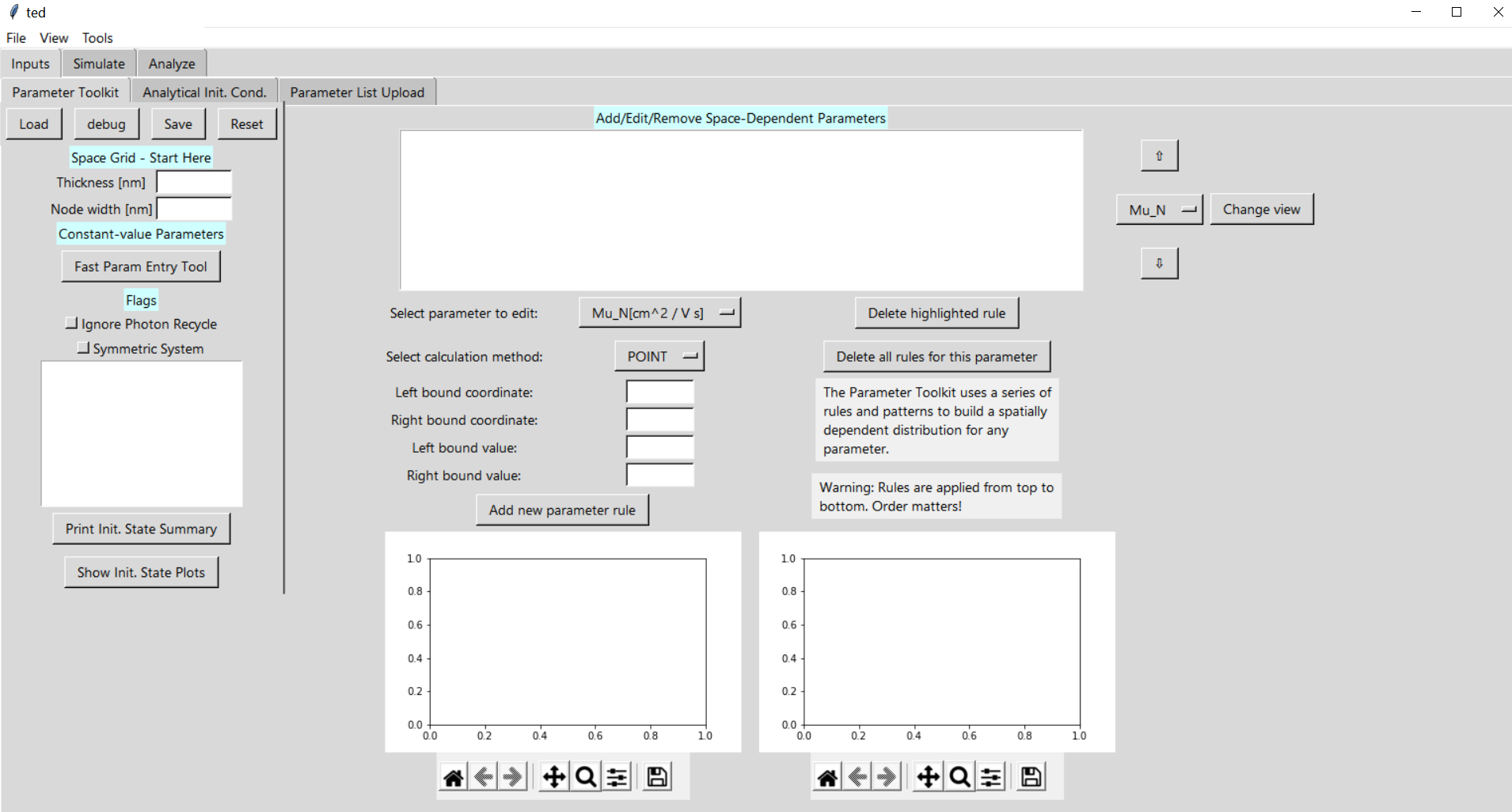


TEDs scans the file **modules.py** for available modules and lists them in the Module Selection window. After selecting a module, TEDs performs some verifications to ensure that the module is correctly implemented, with all issues printed to console. After verification, TEDs then checks for the presence 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 from TEDs be stored in these directories – they are the first place TEDs will search for files, and TEDs may not be able to locate files stored elsewhere.

## Familiarizing with the Layout

After choosing a module, the following interface is displayed.



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.



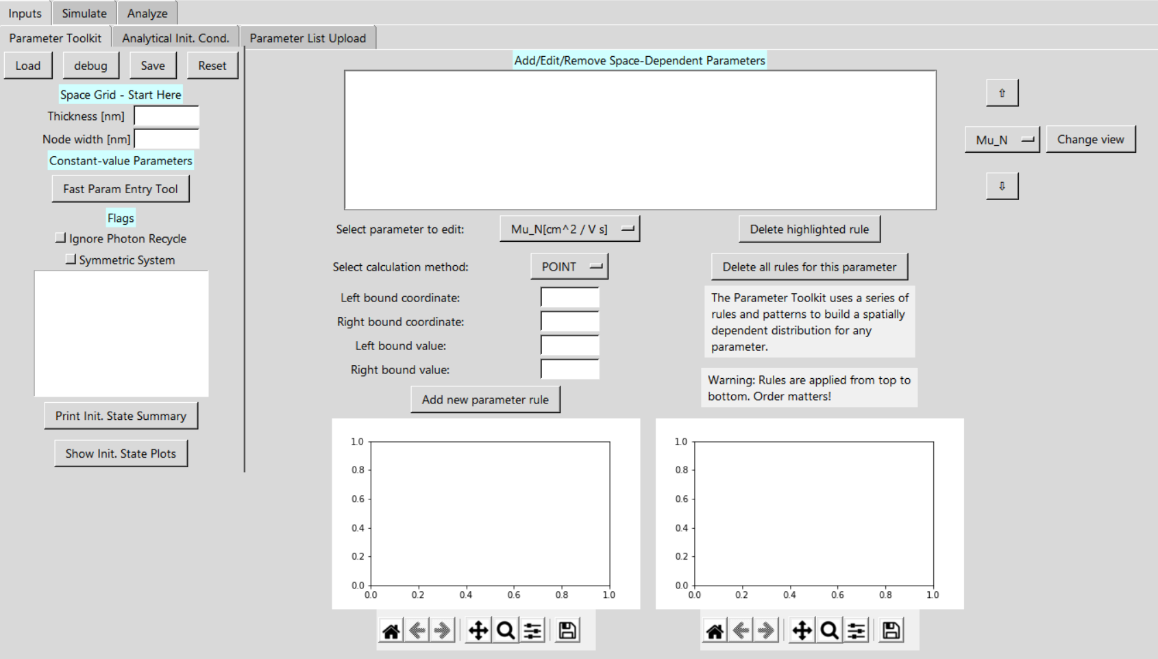
The top left corner also contains three tabs—Inputs, Simulate, and Analyze, which correspond to the three main simulation stages of preparing an initial condition and parameter set, simulating the evolution of the initial condition over time, and viewing and analyzing the results. Each of the sub-interfaces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

# The Inputs Tab - Creating Initial Condition Files

## Overview

The inputs tab contains the following major components:

**1**



**10**

**7**

**6**

**5**

**4**

**3**

**2**

**8**

**9**

1. **Initial Condition Subtabs**

The Inputs Tab itself contains three subtabs – Parameter Toolkit, Parameter List Upload, and Analytical Initial Condition. While initial condition and parameter distributions are zero by default, these three subtabs correspond to three methods available to set values in the distributions. 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 initial state files, discussed further in the sections *Importing Initial Condition Files* and *Exporting Initial State Files*.

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

Once a parameter is entered, 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.

3. **Space Grid Entry Area**

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

4. **Fast Parameter Entry Tool**

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

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.

5. **Flags Area**

Module-specific flags can be toggled here. The “symmetric system” flag is unique – any module which includes this flag will, when this flag is activated, mirror systems that range from position 0 to L over the range 0 to -L.

6. **Status Window**

This box displays status and error messages associated with managing the initial conditions. If a problem occurs while importing, editing, or exporting an initial condition, the issue will be named in this box.

7. **State Summary Buttons**

These buttons can be used to view a list or plots of the current parameter values and can be used to verify an initial 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 *Analytical Initial Conditions, Parameter Toolkit,* and *Parameter List Upload.*

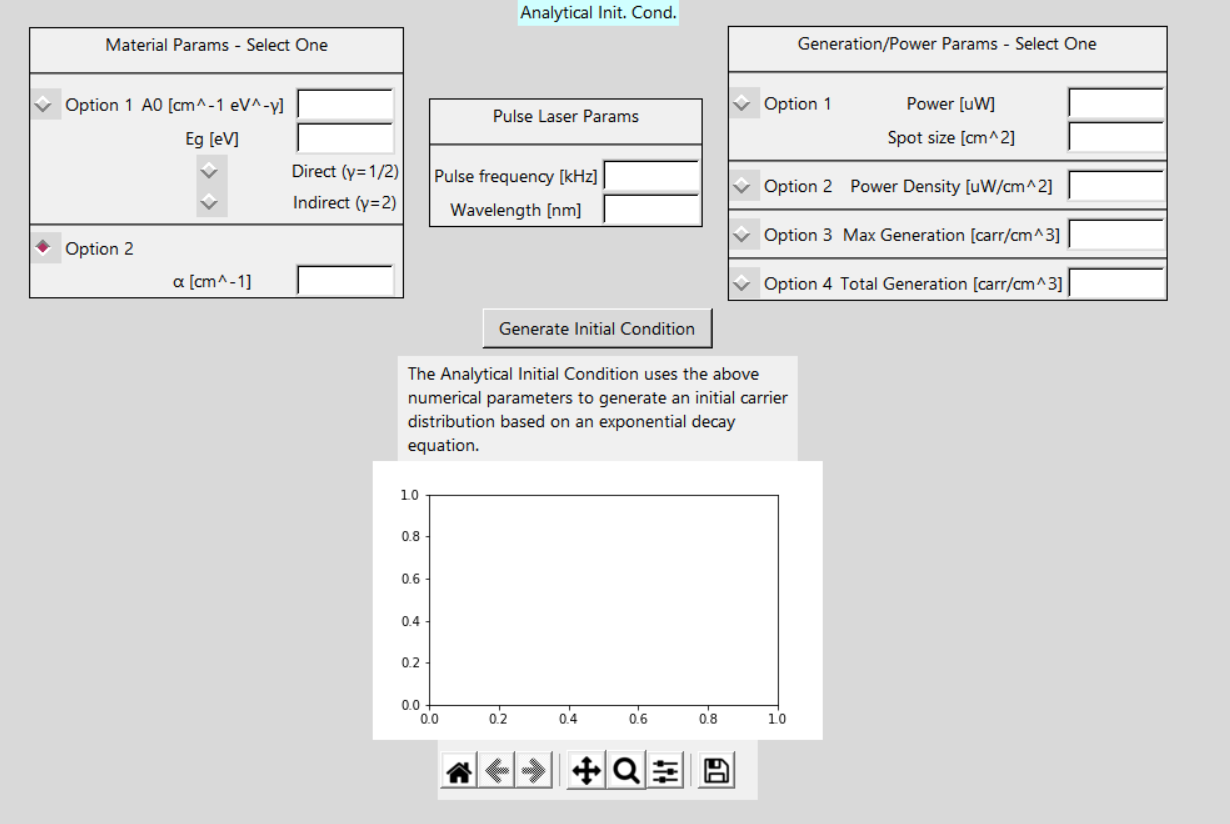
9.  **Plots**

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

## Analytical Initial Conditions (AIC)



The Analytical Initial Condition is exclusive to the Nanowire module and 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 space point across the entire length of the nanowire using the following equations:

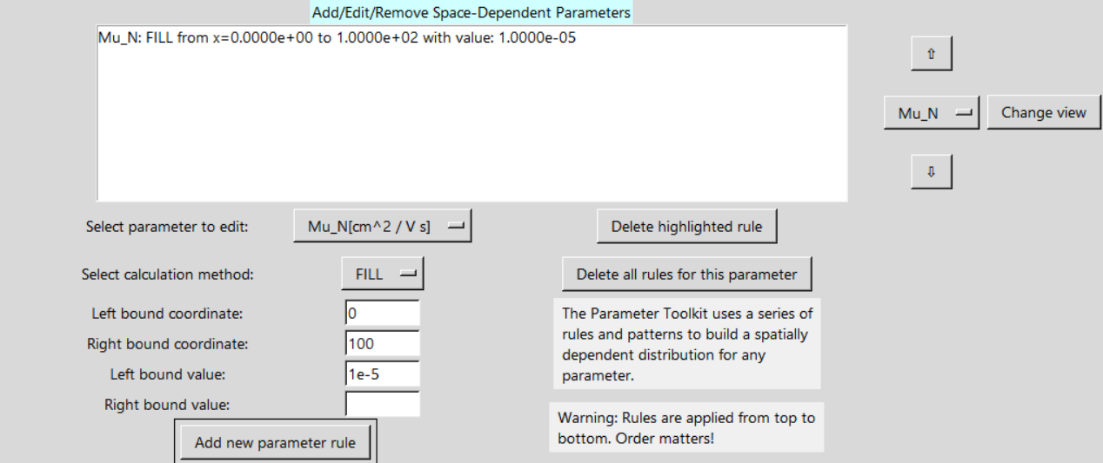
First, the material parameter may be calculated from the below or entered directly.

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

TEDs automatically performs all unit conversions needed to obtain ΔN and ΔP in carriers/nm3 from values inputted in the listed units.

## 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:



**5**1. **Initial Condition Subtabs**

The Inputs Tab itself contains three subtabs – Analytical Initial Condition, Heuristic Initial Condition, and Explicit Initial Condition. These three subtabs correspond to three methods available to generate initial condition distributions and affect the layout presented in [6].

2. **Save and Load Buttons**

These are used to import and export initial condition files, discussed further in the sections *Importing Initial Condition Files* and *Exporting Initial Condition Files*.

3. **Parameter Input Boxes**

The list of system parameters, coefficients, and grid point spacing required by TEDs should be entered in these boxes. The values of the units entered must have the units indicated by the labels to the left of each box.

4. **Status Window**

This box displays status and error messages associated with managing the initial conditions. If a problem occurs while importing, editing, or exporting an initial condition, the issue will be named in this box.

**4**

**3**

**2**

**1**

1. **Rule List Window**

When a new parameter rule is added, a brief description if 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 variable. Only one variable’s rules are displayed at a time – see [3] for changing which variable is displayed.

2. **Reordering Buttons**

Rules in [1] can be selected with a click and can be reordered in the list using these buttons. As mentioned, the order of which heuristics appear in the list is the order in which TEDs will apply them to the distribution.

3. **Variable Selection**

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

4. **Rule Input Boxes**

These items are used to, for a new rule, specify the variable 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*.

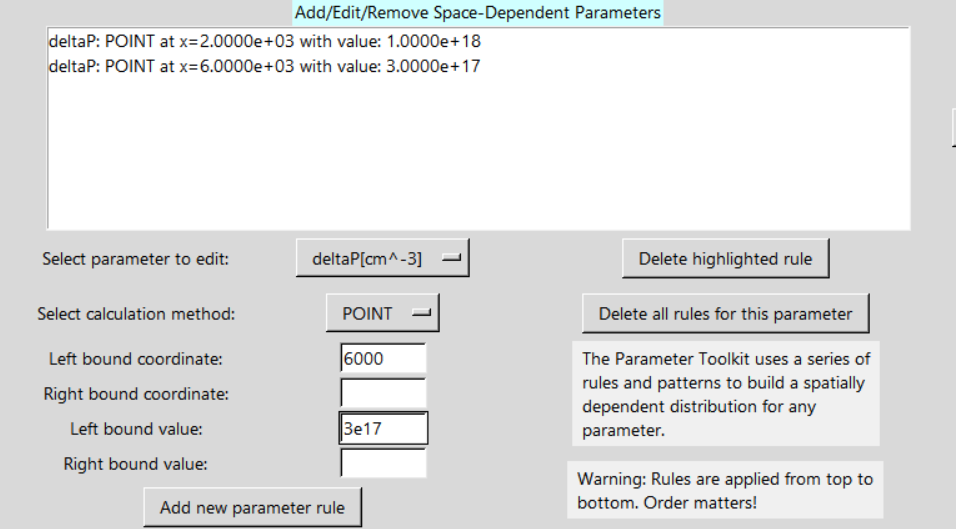
5. **Deletion Buttons**

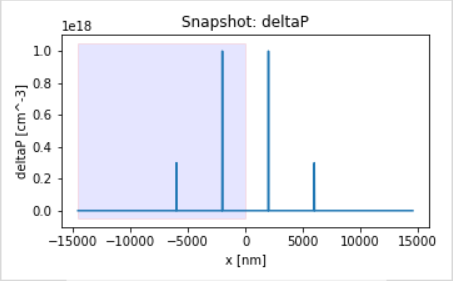
These buttons are used to delete either the selected rule or clear all rules for the currently displayed variable.

## Parameter Toolkit Mathematical Methods

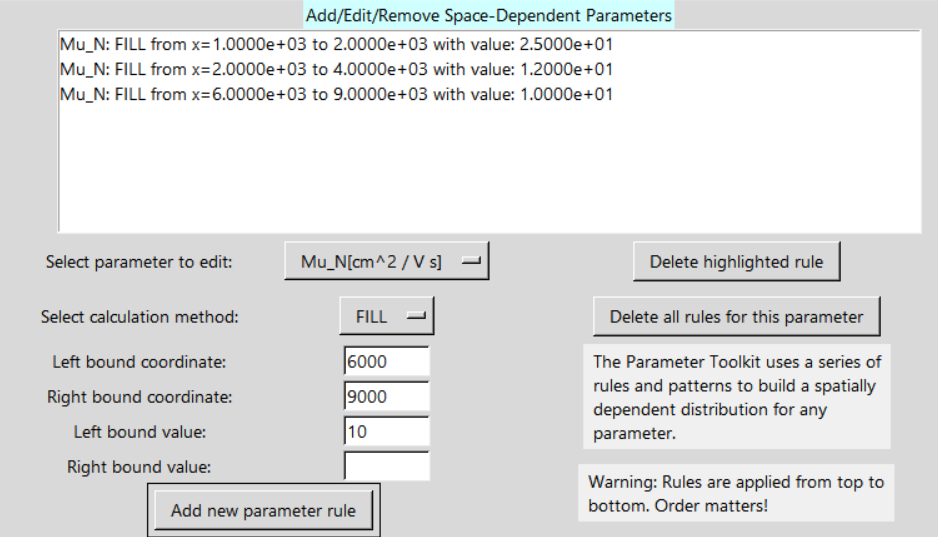
Four mathematical methods are available for the parameter toolkit – POINT, FILL, LINE, and EXP.

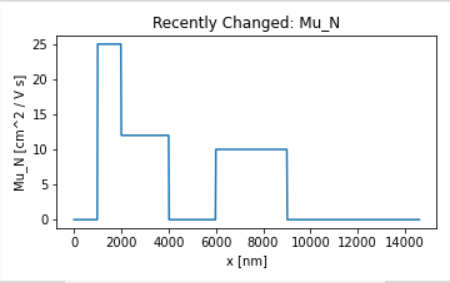
The POINT method is by far the most straightforward, assigning the Left Bound Value to the selected variable’s initial distribution at the location specified by Left Bound Coordinate. In the following example, this method is applied twice to create a very pointy ΔP initial distribution.

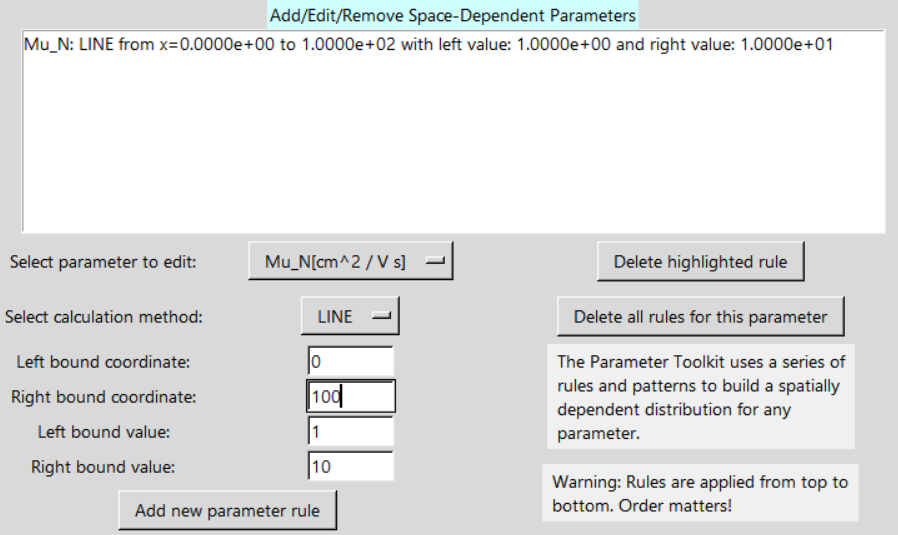


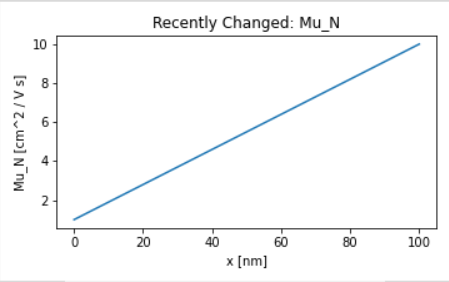


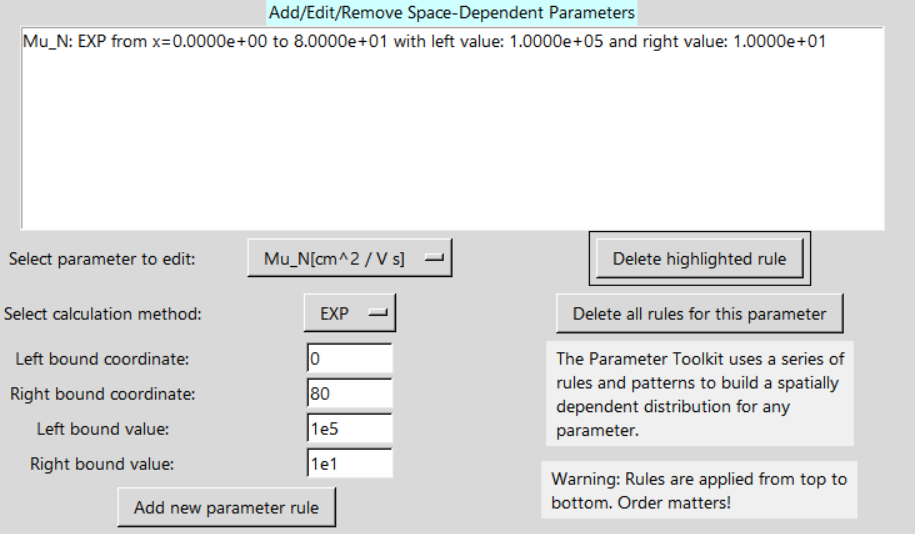
Note that in this example the “symmetric system” flag is activated – which causes the plots to display the symmetric half of this nanowire.

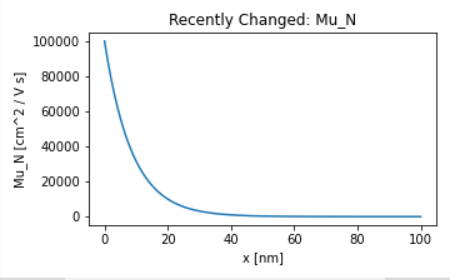
The FILL method fills every space point from the Left Bound Coordinate to the Right Bound Coordinate with the Left Bound Value. This method is useful for specifying regions with constant values and is a special case of the LINE method.

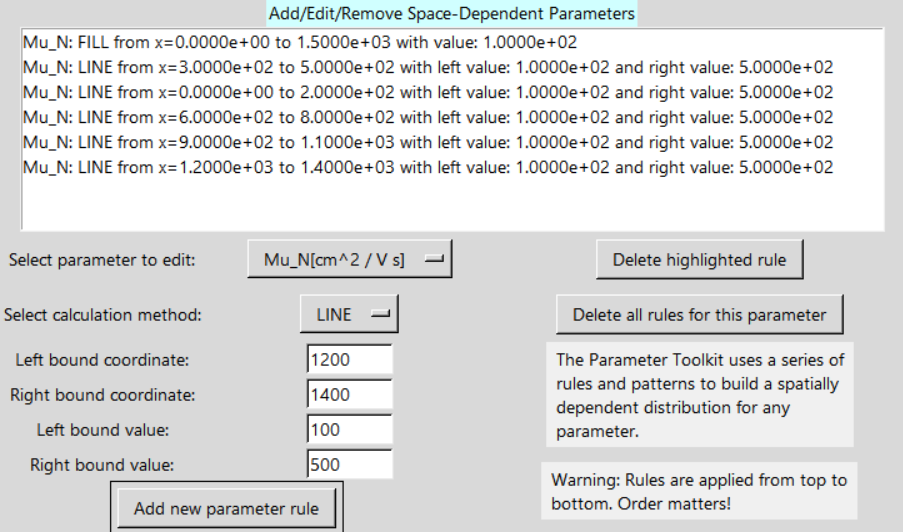


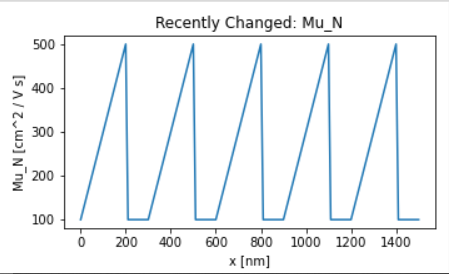
The LINE method assigns the Left Bound Value to the Left Bound Coordinate, assigns the Right Bound Value to the Right Bound Coordinate, and performs linear interpolation to assign values to all intermediate space points. Each intermediate point differs from its neighbors by a common difference value.



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.



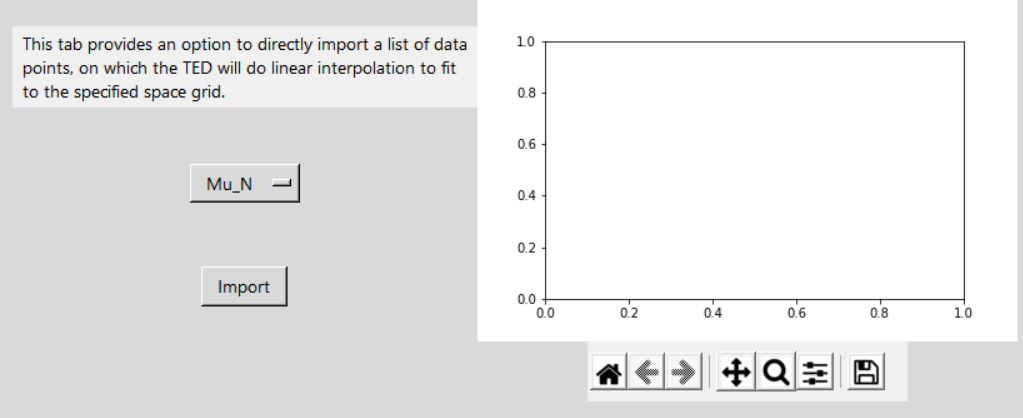
These four methods can be combined to set up very complex distributions.



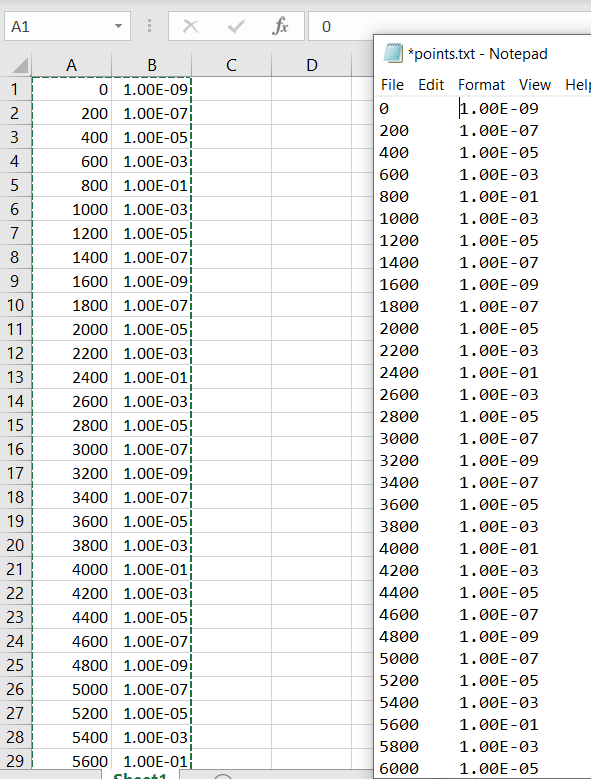
## Parameter List Upload

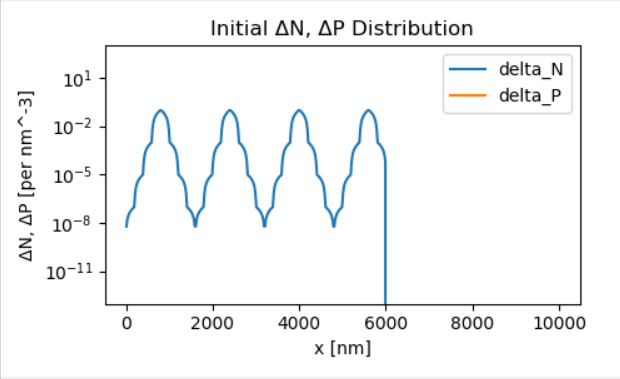
Copying from Microsoft Excel causes the data to be pasted into most text editors as tab-separated columns – a fast way to create properly formatted EIC files!

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.



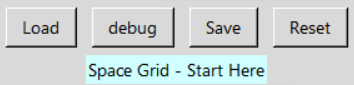
In the following example, the EIC file “points.txt” is applied to ΔN over the range z=0 nm to z=6000 nm.





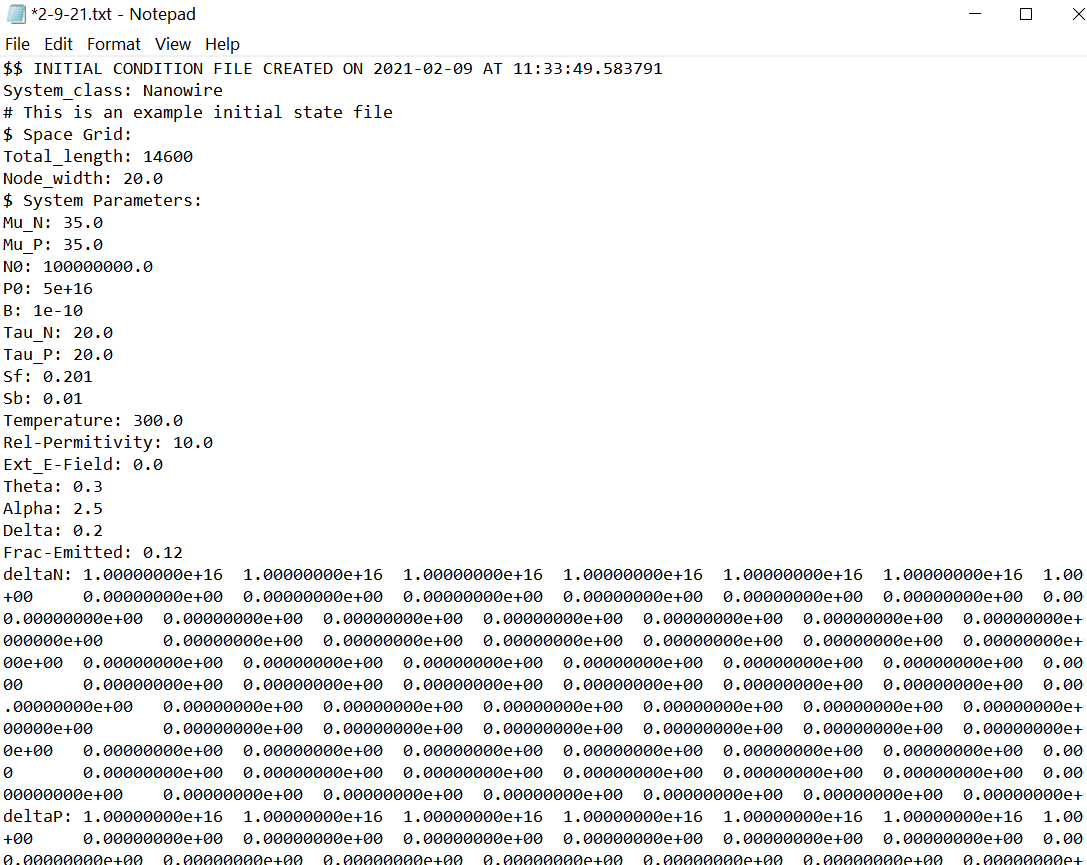
## Saving and Loading Initial State Files

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



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

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



The order of system parameters seen in this example is the order in which TEDs will save them, but TEDs can load and read these in any order. An ISF like this one whose System Parameters section starts with Temperature and ends with Theta, for instance, is perfectly compatible.

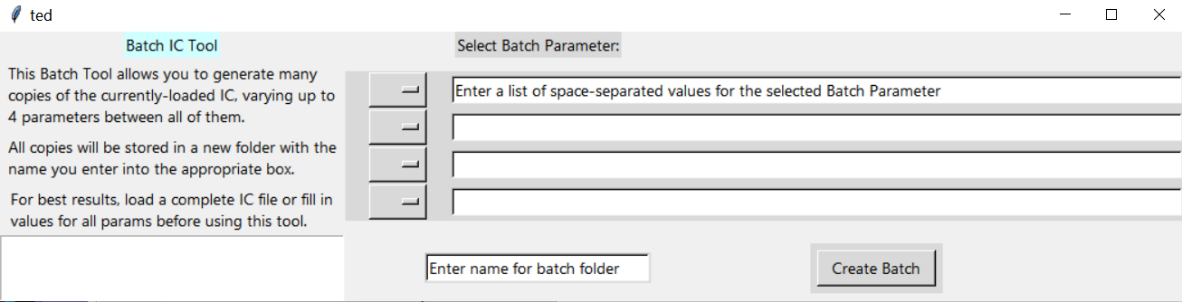
The first line of an ISF contains a timestamp of when the file was created; immediately following that is the “System class” identifier, which describes which module the file is for. The “$ Space Grid” section contains the length and space stepsize of the space grid. The next section, “$ System Parameters”, represents the lists of values that form the parameter distribution at every data point. Constant values across the space grid are saved as a single value while space-dependent distributions, as with deltaN and deltaP in the above example, are saved as a tab-separated list.

Tab-separated lists are also directly pastable from Excel.

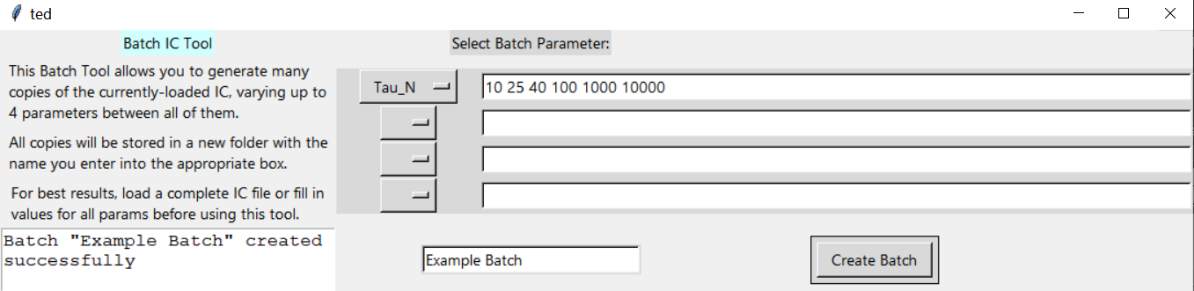
User comments may be entered below the “System class” identifier with the “#” symbol, as shown in the above example.

## The Batch Initial State File Tool

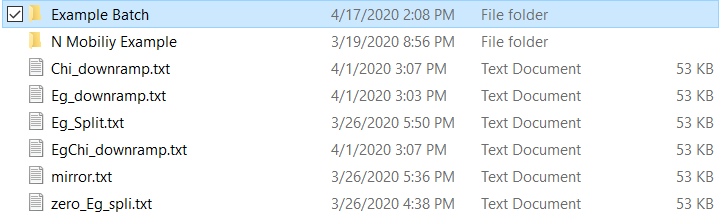
In many situations, such as sensitivity analyses, it is useful to generate many initial condition files over a rectangular parameter space. In the “Tools” menu, TEDs offers the Batch Initial State File Tool, which provides a fast method to generate many such copies based on an existing initial state.

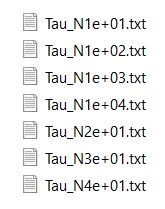


In the following example, the Batch Initial State File Tool is used to generate copies of “2-9-21.txt” from *Saving and Loading Initial Conditions* with varying values of the parameter Tau\_N: 10, 25, 40, 100, 1 000, and 10 000.



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

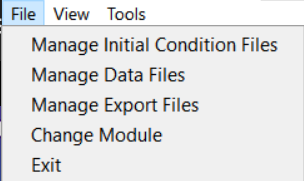




Up to four parameters may be varied at a time.

## Removing Files

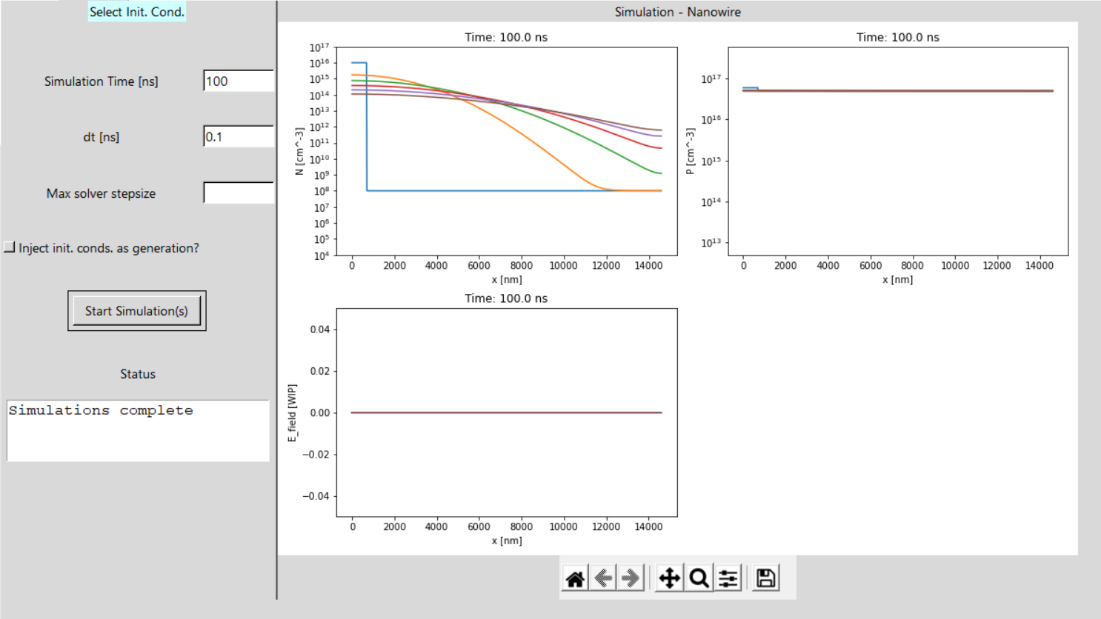
In the top left corner menu under “File”, TEDs provides a series of shortcuts to each of the default directories.



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

# Running Simulations

The simulate tab contains the following major components:



**1**

**4**

**3**

**2**

1. **Simulation Parameters and Options**

This area contains input boxes in which the total time over which the system should be modeled and the time step size this total should be partitioned into, as well as options to modify the behavior of the simulation. System-specific flags are specified in the Initial tab.

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

If the “Ignore Photon Recycle” is active for a Nanowire, TEDs will neglect charge carrier regeneration due to photons propagating down the nanowire. This approximation makes the simulation slightly less accurate at early times but greatly increases the speed of the calculations overall.

All other behaviors, such as photon escape and photon propagation *without* charge carrier regeneration, will remain active.

If “Inject init. conds. as generation” is checked, TEDs will treat the initial condition distributions as a constantly applied external stimulation rather than an initial burst. The initial condition distributions will not only be the starting state of the nanowire, for example, but the initial ΔN and ΔP will be added to the nanowire every time step.

2. **Calculate Button**

When this button is clicked, a prompt opens for selecting previously saved ISFs. 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 in the “Data” directory specified in config.txt containing the results of the simulation. The name of this folder is based on the name of the ISF used to run the simulation.

3. **Status Window**

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

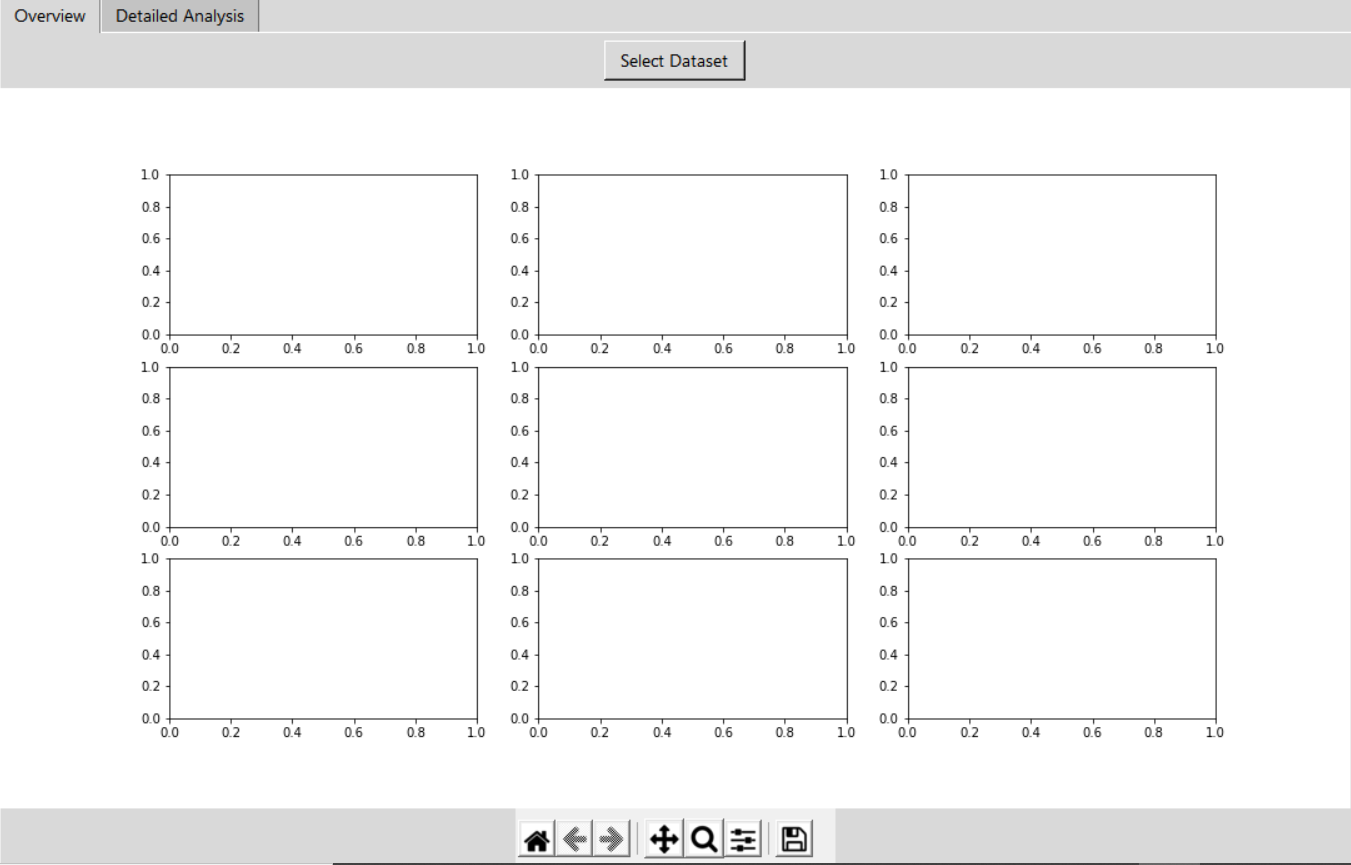
4. **Results Plots**

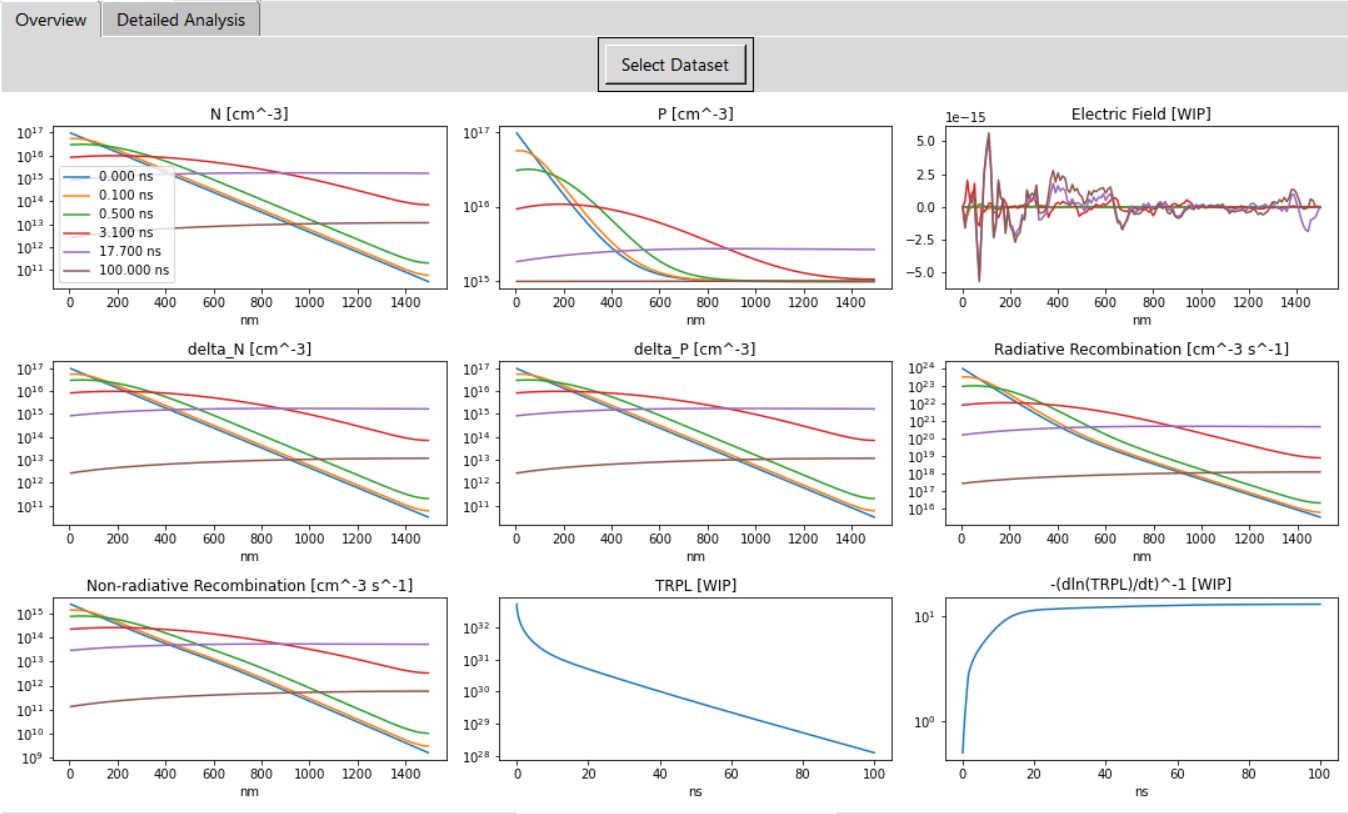
These plots display snapshots of time steps taken during the simulation, a useful first glance of how the nanowire is evolving over time. As with the initial tab plots, each of these plots has a toolbar for basic resizing and exporting.

# 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 directory.

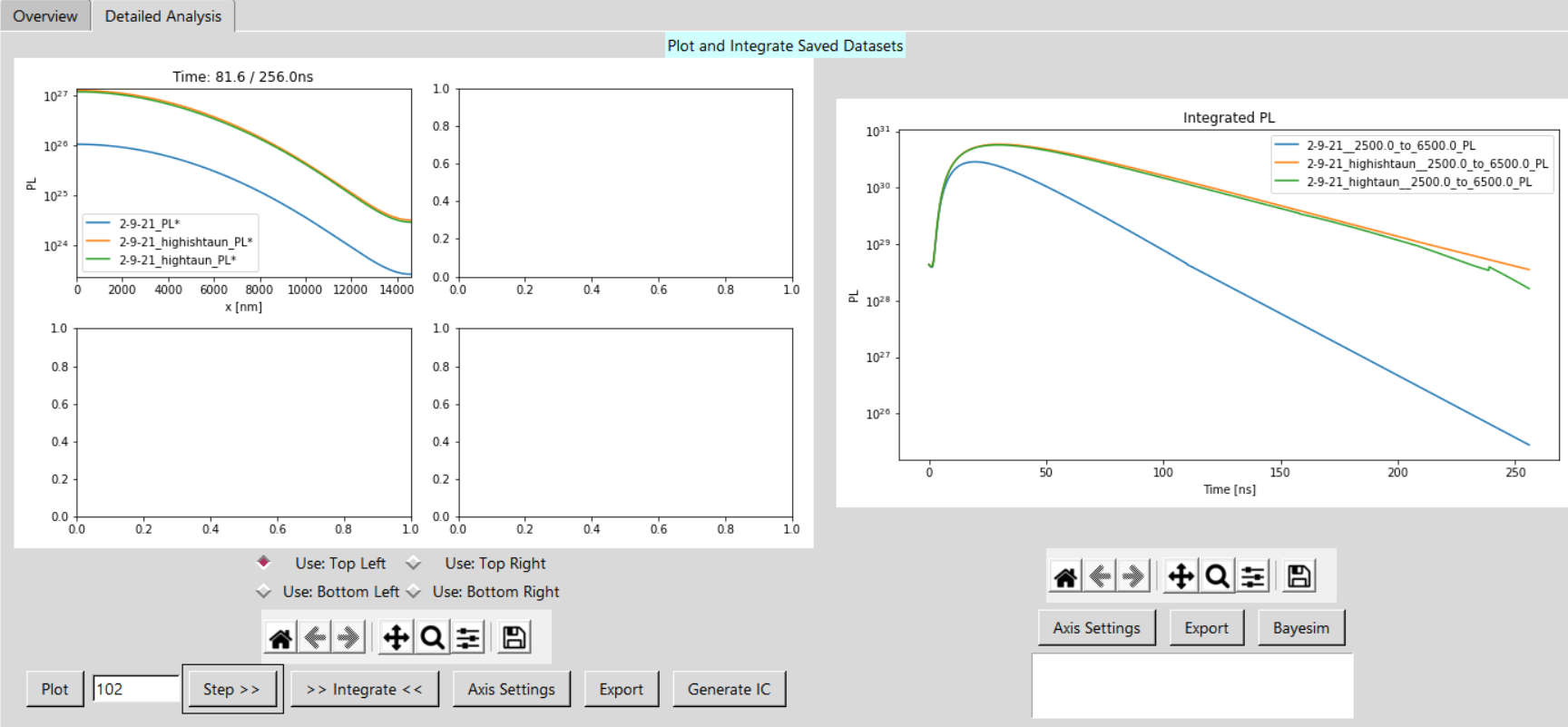




To view other timesteps or calculated TRPL over other ranges, for example, the Detailed Analysis should be used.

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.

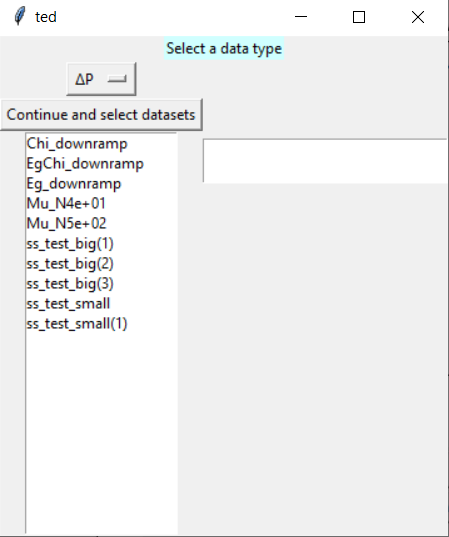
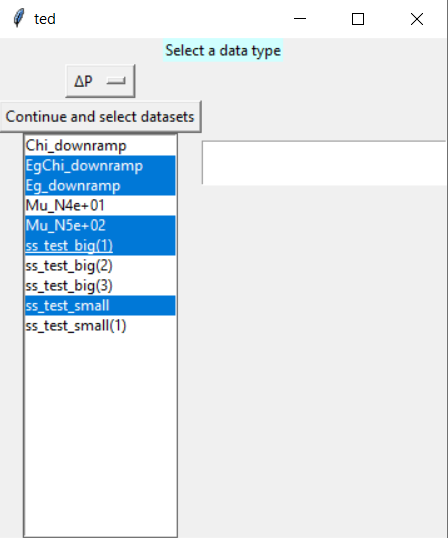
The defining feature of this tab is that each plot is equipped with a more detailed toolbar, and the workings of each button are covered in the following sections.





## 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.

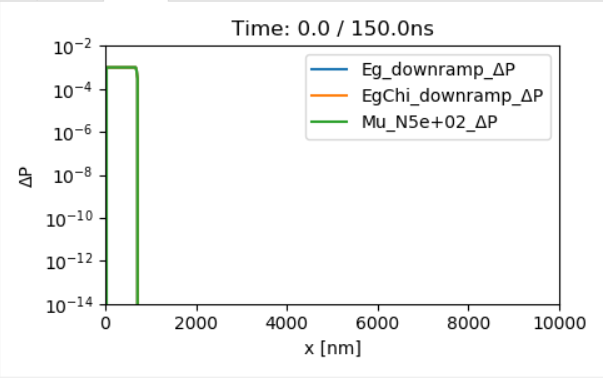
 

To select a single dataset, click its name in the box.

To select multiple datasets, hold the **Ctrl** key and drag the cursor over every desired dataset.

Excel users will notice that TEDs uses the same multiple cell selection scheme as Excel.

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.



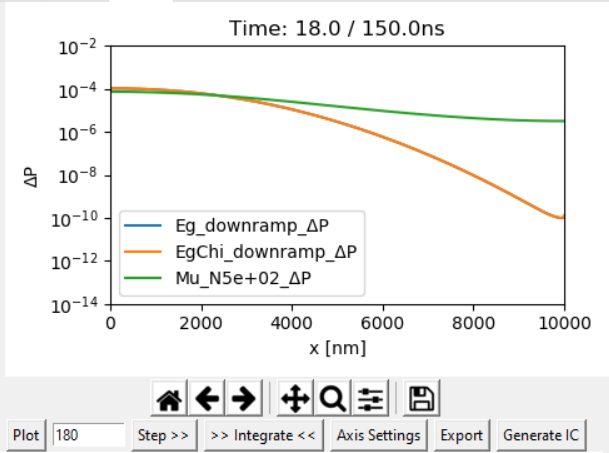
Datasets with different **space** step sizes, however, are compatible with one another.

One major limitation, as demonstrated above by the fact that only three of the five 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.

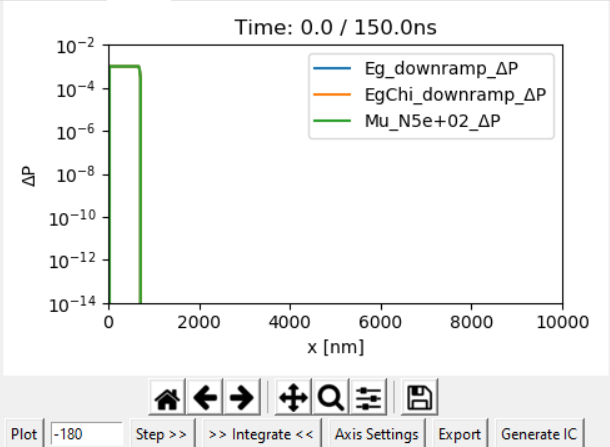
## Stepping Datasets Through Time

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 0.1 ns, for example, entering 180 will advance the datasets by 18.0 ns. each time the “Step” button is clicked.

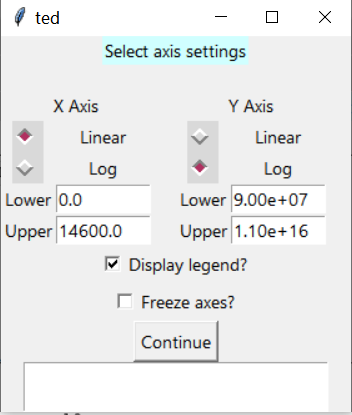


Stepping with a negative number of time steps will cause TEDs to move the datasets **backward** that many time steps.



## Changing Axis Settings

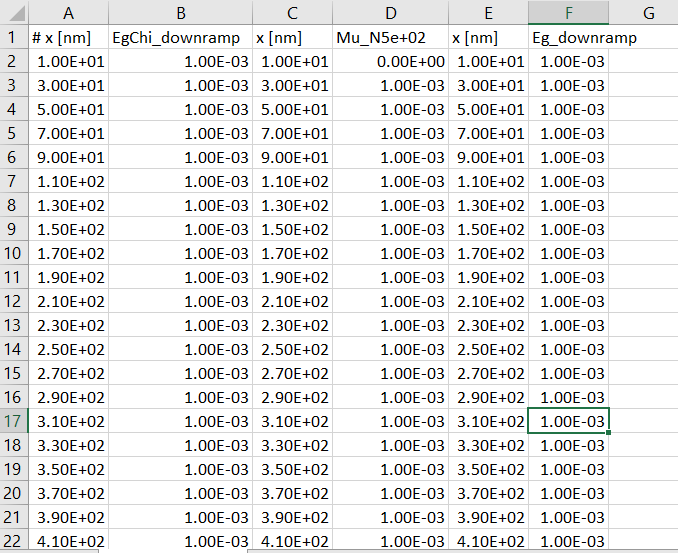
The “Axis Settings” button opens a popup to change the lower bounds, upper bounds, and scaling type of the x and y axes. Options to toggle the legend visibility or freeze the axes are also available.



TEDs will attempt to determine the best axis range and scale (linear or log) based on the values spanned by the data. Sometimes this does not work. In that case the “Freeze Axes” option can be used to specify the axes manually.

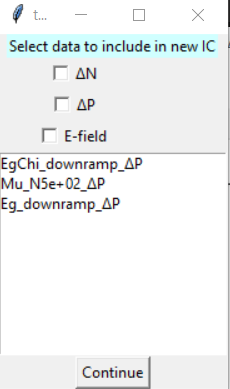
## Exporting Data

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 – 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.



## Carrying Existing Data into New Initial Conditions

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



Unselected variables will be filled in with the default value – zero.

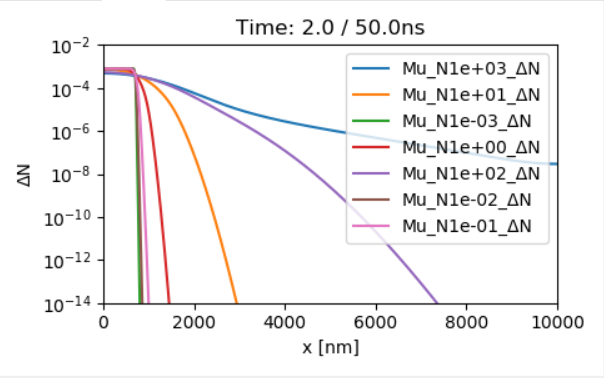
First, the checkboxes are used to indicate which variables should be used to generate initial condition distributions. The selection box, which functions like the “Plot” button’s, is then used to indicate which datasets ISFs should be generated for. When “Continue” is clicked, one ISF will be created with the selected variables for each dataset and prompts will appear to name the ISFs.

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

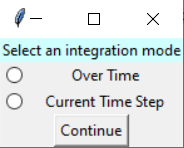
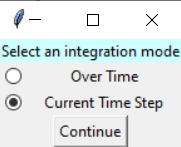
## Integration

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 the horizontal axis is not used for time 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.

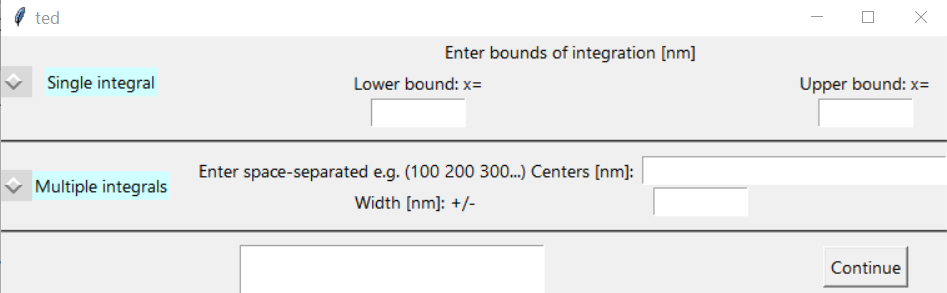
For example, we may want to examine how the carrier density near the front of the nanowire is affected by the negative charge carrier mobility “Mu\_N”, but a plot of the spatial distributions can be quite chaotic.



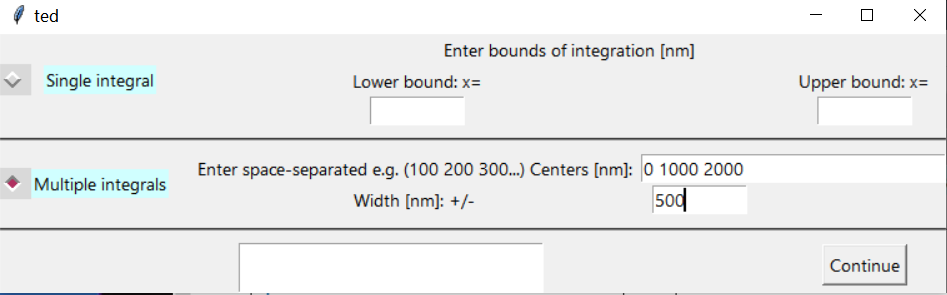
Upon clicking “Integrate”, the following popup appears for selecting the integration mode.

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



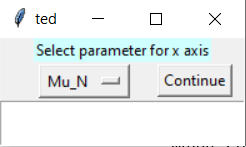
The “Single integral” option performs one integral per dataset over the specified bounds. The “Multiple integrals” performs a set of integrals per dataset, each at a specified centerpoint with given width. These integrals will be cut off at the boundaries if the centerpoint and width would normally take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



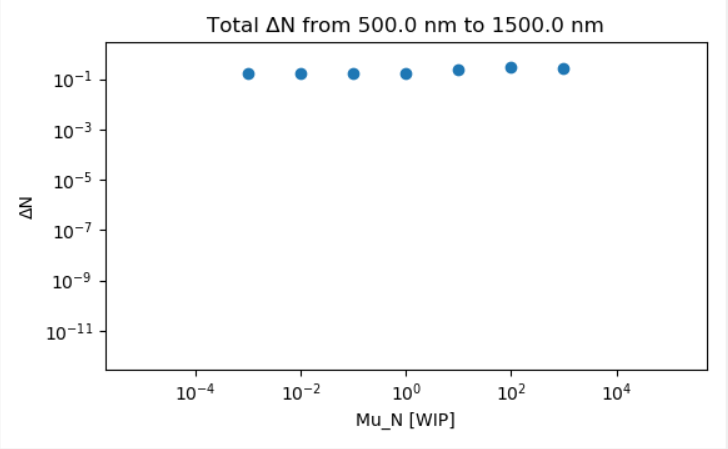
In this example, integrals over [0,500], [500,1500], and [1500,2500] will be performed.

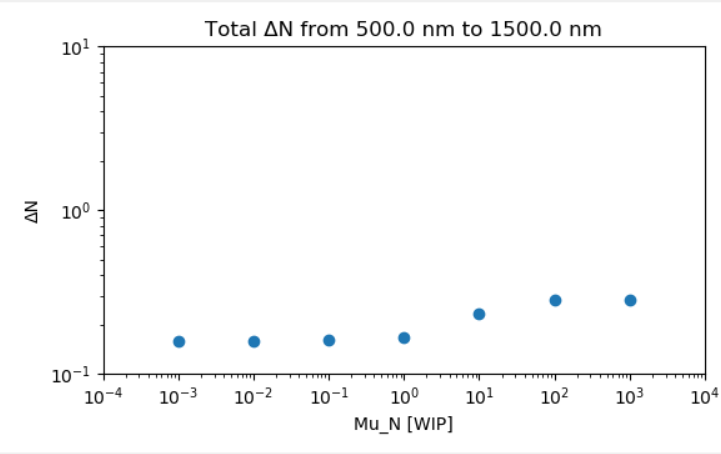
If the “symmetric system” flag was active during a simulation, the integration can pass into the symmetric “other half” – for example, the previous will instead span over [-500, 500], [500,1500], and [1500,2500].

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

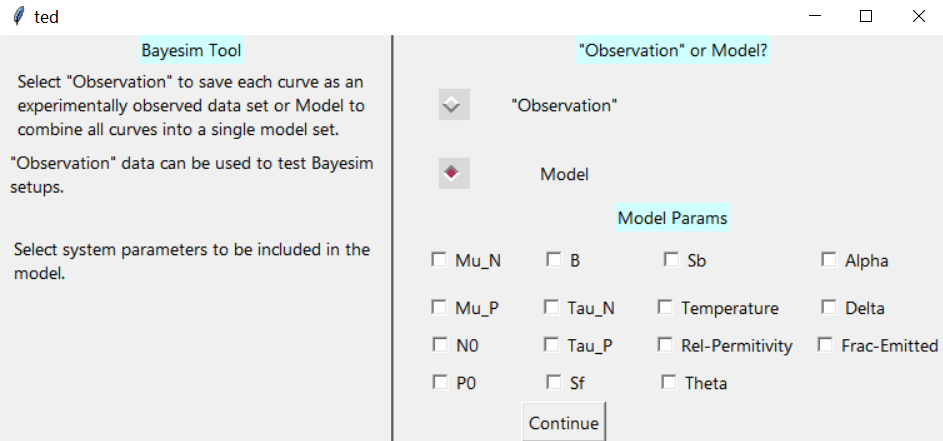


The following plot is generated, but adjusting the vertical axis using the “Axis Settings” should prove helpful.





## Experimental Feature – Bayesim Export



For compatibility with the Bayesim package, the Bayesim tool allows export of integrated data as either a Bayesim observation file or model object. The observation file saves only the integrated values and can be used to give Bayesim fake experimental data for testing purposes. The model option generates a Bayesim model using the selected parameters and is best used in conjunction with the Batch ISF tool to create data for a parameter grid.

# Adding a Custom Module

Note: HagesLab is working on simplifying this procedure, but generally, intermediate Python knowledge is recommended here!

By extending the OneD\_Model template class in the file **modules.py**, custom models can be created and simulated with TEDs. A complete module must implement the following functions with the exact arguments specified in the tempate:

* \_\_init\_\_()
* calc\_inits()
* simulate()
* get\_overview\_analysis()
* prep\_dataset()
* get\_IC\_carry()

Once complete, the module and its display name must be added to the mod\_list() at the top of **modules.py**. The new module can then be selected when TEDs is started.

The Nanowire and HeatPlate modules are provided as examples.

## \_\_init\_\_()

This function sets up all informational variables associated with the module and informs TEDs of whether variables are parameters or outputs. 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.
* length\_unit
  + This is a string specifying the length unit (e.g. nm, m, km) the space grid is calculated in. This unit is for display purposes only (i.e. not used in calculations), but the length and space node width are assumed to be in the same unit.
* 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 display units of the parameter, whether it should be calculated at a space node center or node edge, and the values of the parameters to be passed to ODEINT.
* flags\_dict
  + This dictionary can be used to specify the {“internal name”:”display name”} association for each system flag. One flag toggle is created in on the Initial tab for each flag in flags\_dict.
* simulation\_outputs\_dict
  + This dictionary specifies which outputs are returned when TEDs simulates a system created from the module and uses Output() objects to store display information and operational flags regarding these outputs. The display information is used to populate plot fields while the is\_calculated and is\_integrated flags are provided for convenience for other parts of the module implementation.
* calculated\_outputs\_dict
  + This dictionary specifies which outputs are calculated from those listed in simulation\_outputs\_dict but not simulated directly. Outputs here must also include a calc\_func attribute which indicates the function used to calculate itself.
* 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. One conversion factor must be defined for each input in param\_dict and each output in simulation\_ and calculated\_outputs\_dict.

## calc\_inits()

This function is used by TEDs to obtain all initial conditions needed by ODEINT and must return a dictionary {“param name”:np.ndarray} containing one 1D array containing the initial value of each space node per output in simulation\_outputs\_dict. Use of the module’s param\_dict, convert\_in\_dict, grid\_x\_node, and grid\_x\_edges is highly recommended.

## simulate()

This function is used by TEDs to call the functions responsible for preprocessing inputs, simulating the data, and 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.

## 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. The module’s is\_calculated and is\_integrated may be helpful to specify which type of processing each output needs.

## prep\_dataset()

This function is used to do preprocessing of data prior to plotting and must return a 1D array of data to be plotted or a 2D array to be integrated. For instance, the PL 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 calc\_func. It is highly recommended that calc\_funcs employ numpy vectorization for compatibility with both 1D and 2D inputs.

## get\_IC\_carry()

This function is used to assign current data values to param\_dict for regenerating ISF 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.