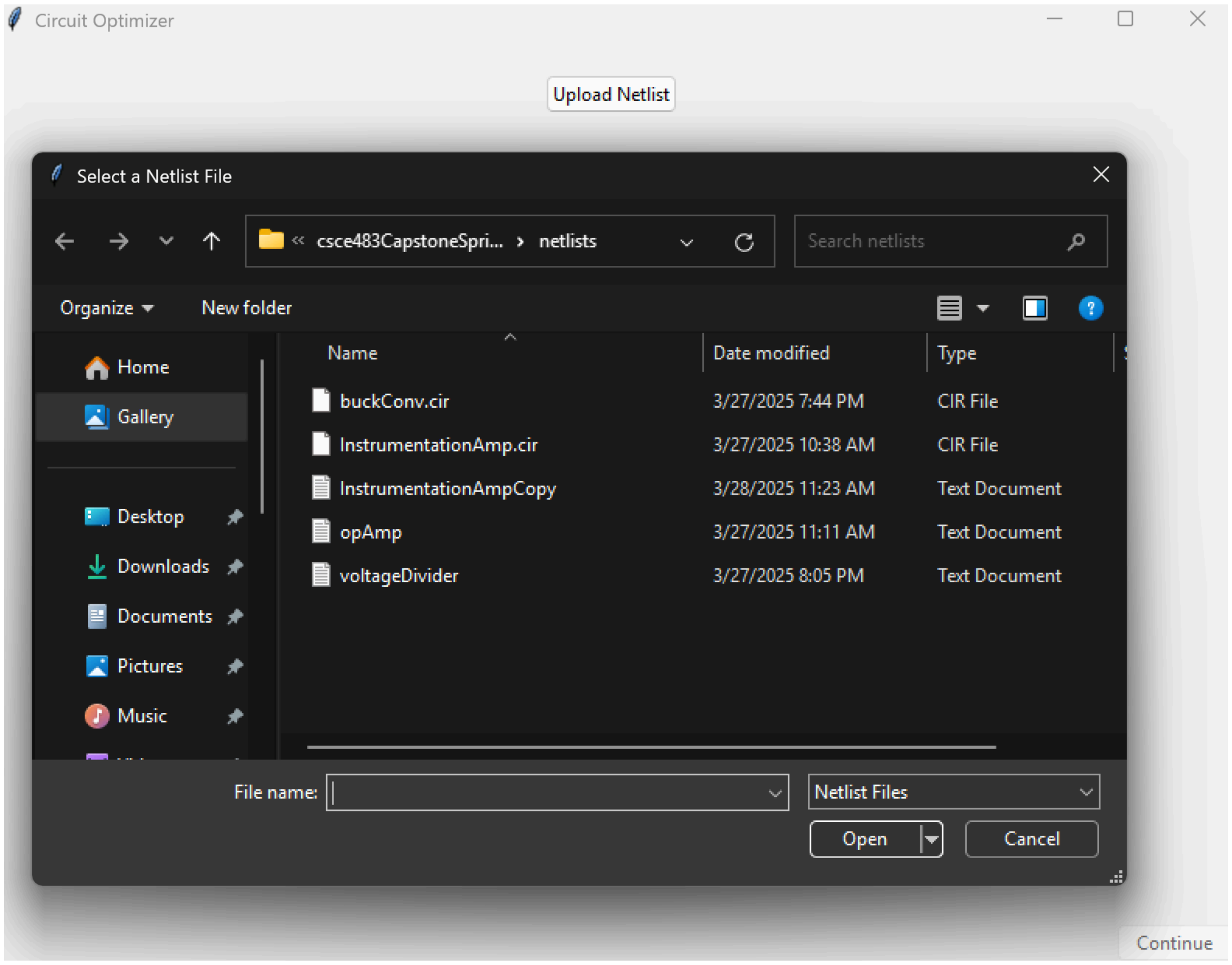
User Manual

Software Installation

Our project consists of a Windows executable. Simply download the file, unzip the project, and run the executable. For our project, you will need a netlist written in Pspice syntax. We recommend using Qucs for a Windows schematic editor that plays nicely with Xyce. You can install Qucs using their website. In addition, your computer will also need to install Xyce. Xyce can be installed on Sandia's website. Xyce also needs to be in your path variable. Simply add the executable Xyce bin file in the path file. In order to check that Xyce is installed correctly, run the command *xyce -v* in the command line and make sure that command runs.

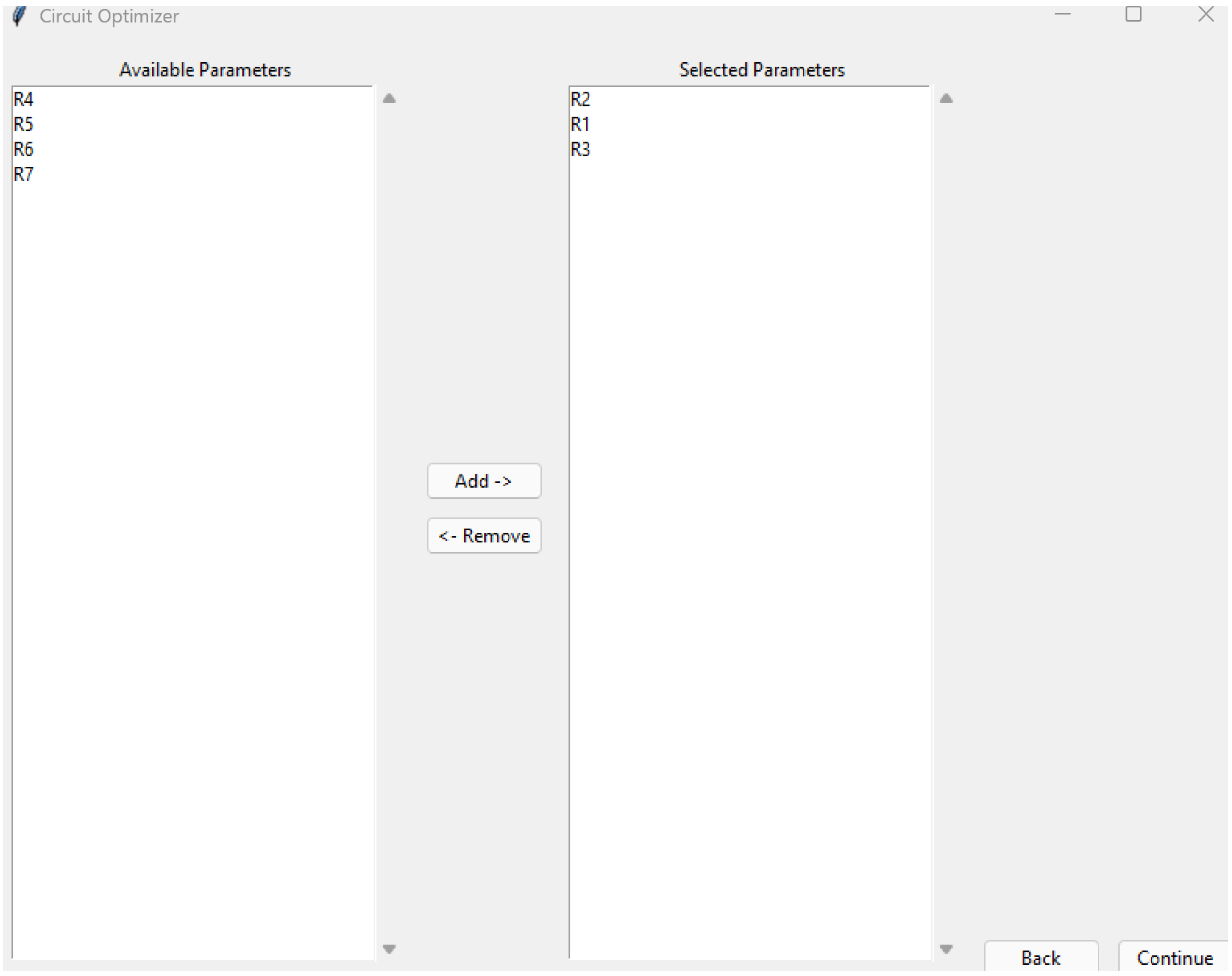
Operation Manual

To use our software first upload the netlist. Our software supports Netlists with syntax that is supported by Xyce. For a more detailed description of what features are supported check out the reference guide. Please do not include .tran or .print commands in your netlist as our software will generate them automatically based on the target function input.

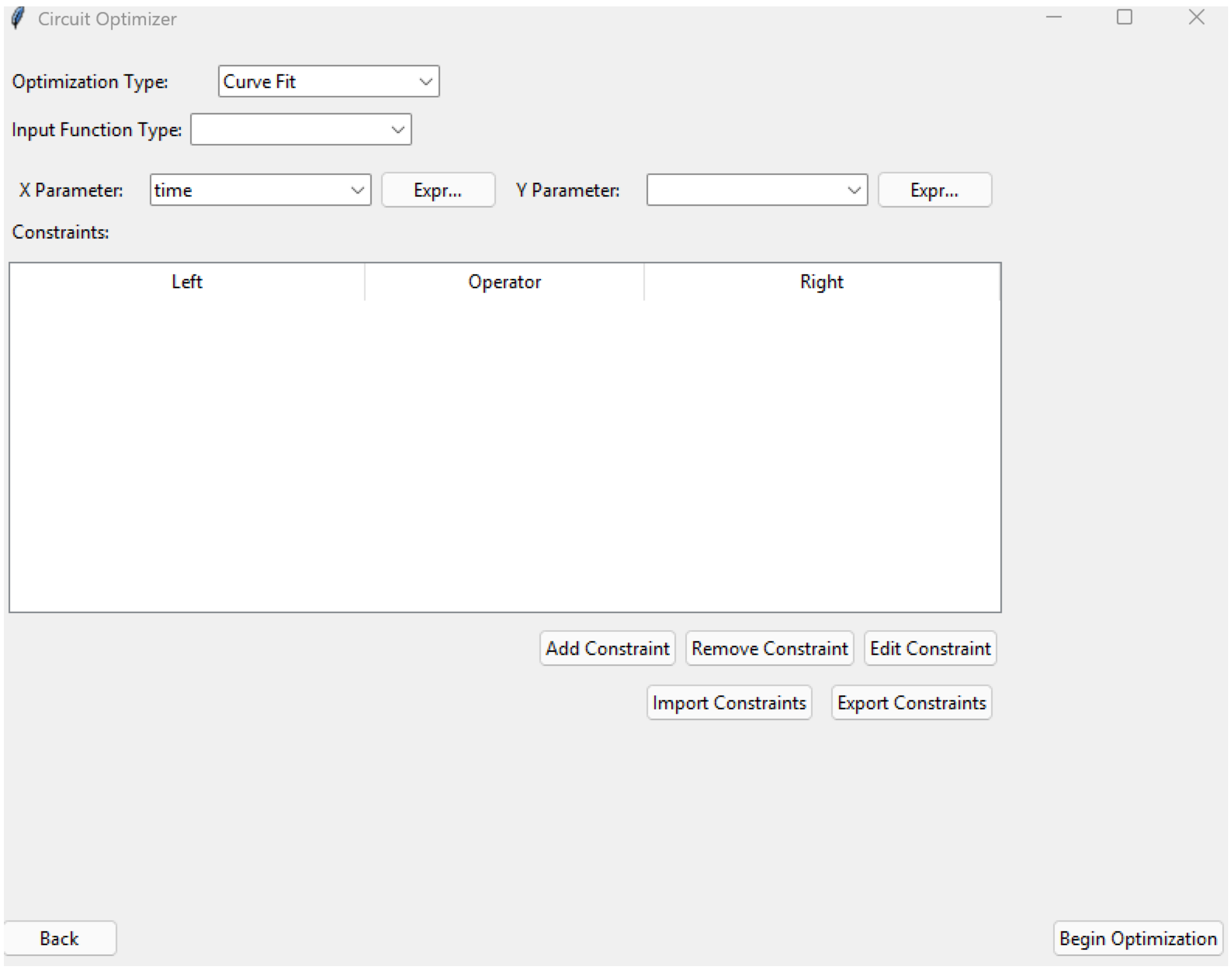


Name	Date modified	Type
buckConv.cir	3/27/2025 7:44 PM	CIR File
InstrumentationAmp.cir	3/27/2025 10:38 AM	CIR File
InstrumentationAmpCopy	3/28/2025 11:23 AM	Text Document
opAmp	3/27/2025 11:11 AM	Text Document
voltageDivider	3/27/2025 8:05 PM	Text Document

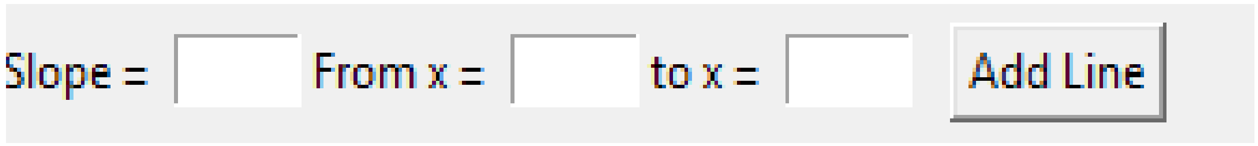
Select a valid netlist file and then press open. Afterward, press the continue button.



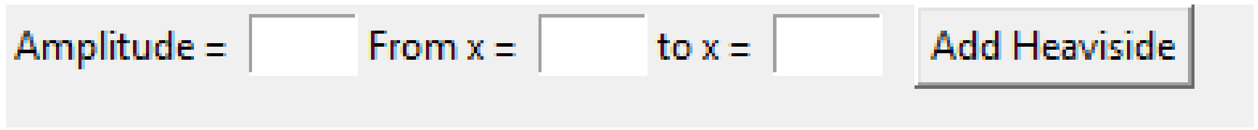
Parameters are R, L, or C values that can be optimized and changed. Select the parameters you want to optimize and move them to the selected parameters section. Parameters that are not selected will be kept at their original value and will not be optimized.



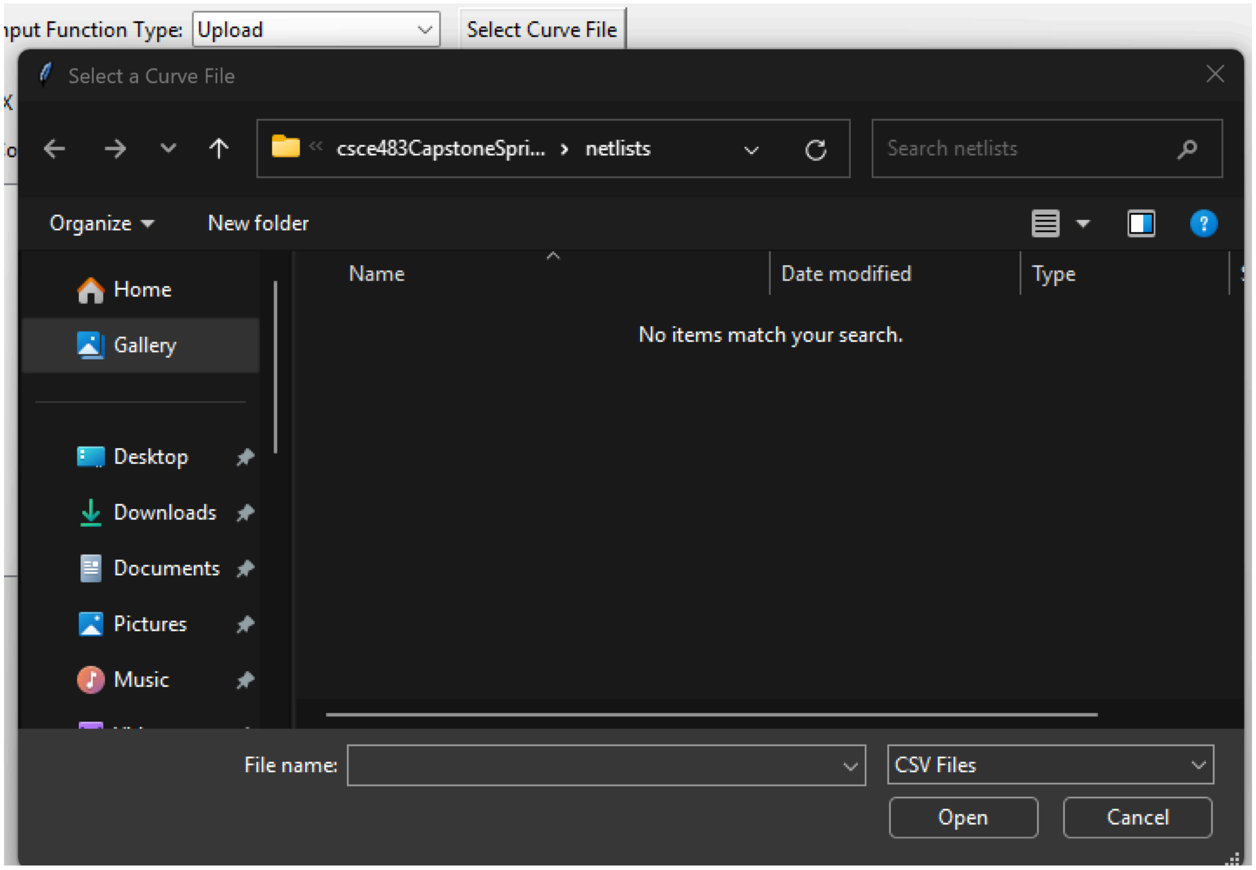
Here you specify the optimization settings. The target function is the type of the function that you want to optimize. The optimizer will try to make your Xyce output equal to the target function. When inputting your target function the specified x values will be the start and stop time of the transient simulation. The units are in seconds. We support three function types: line, heaviside, and custom function.



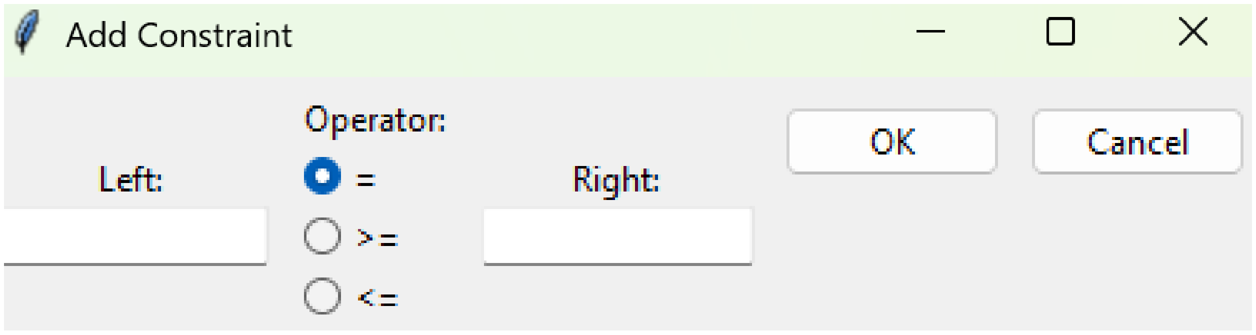
For the line type input the slope as well as the start and stop value for the line and then press Add Line.



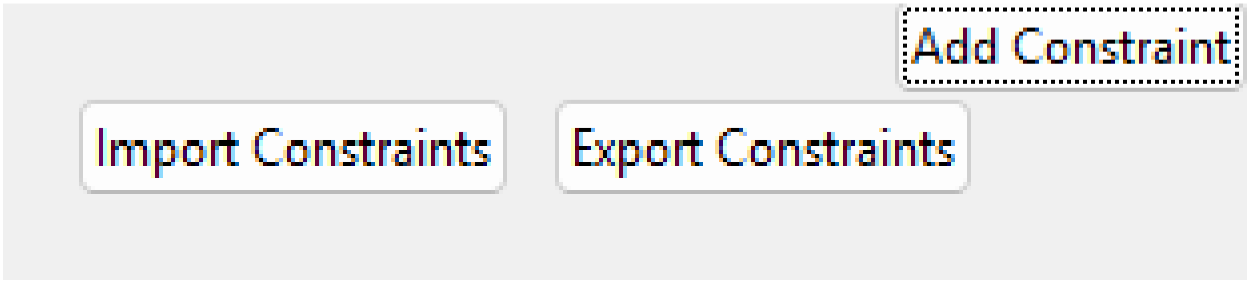
For the Heaviside set the Amplitude, start and stop values, and then press the Add Heaviside button.



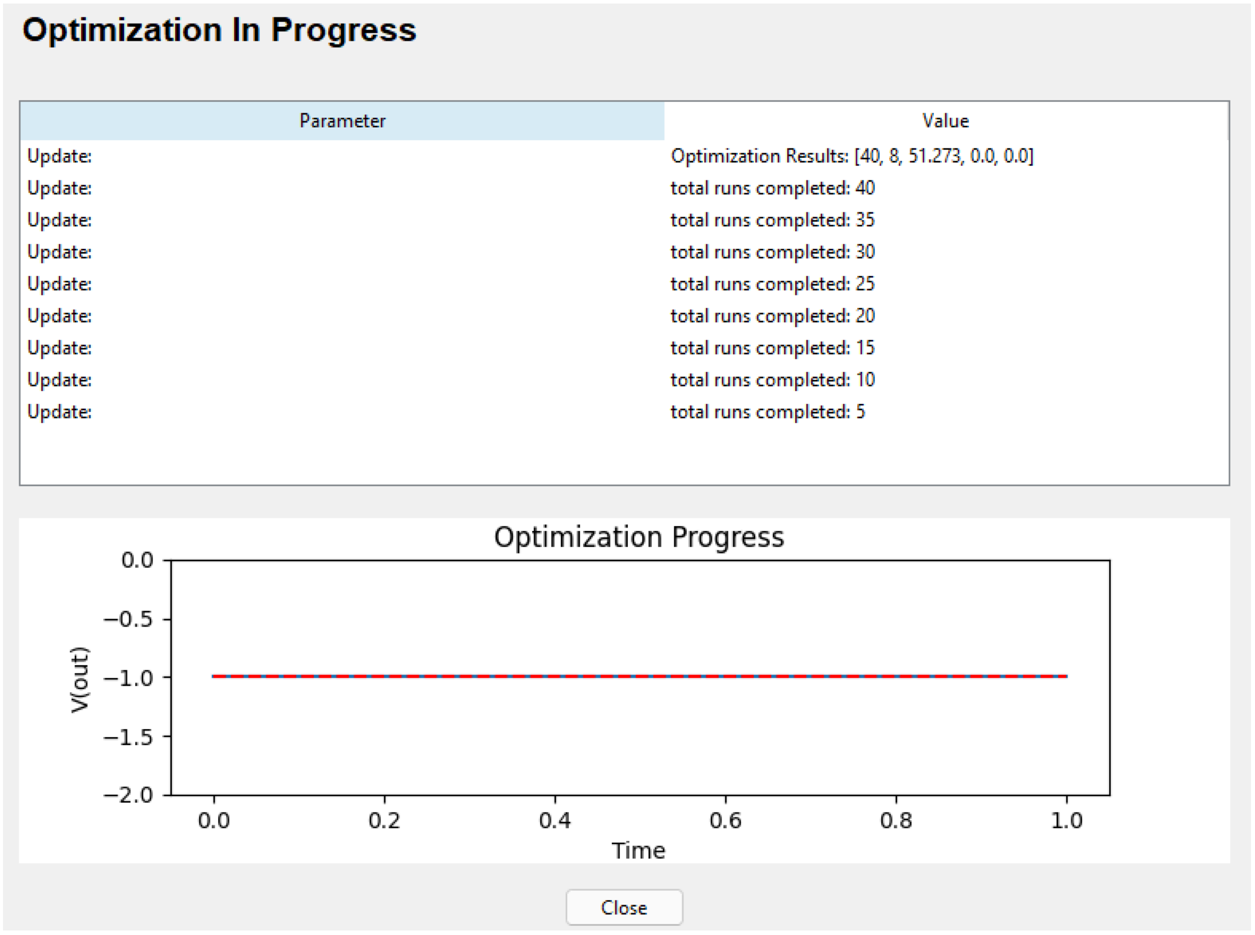
For Custom, press the upload button and then select a CSV file that has a function you want to optimize with. The Y-parameter should be the voltage that you want to optimize the function to. The units for the x axis will be seconds.



Press Add constraint to add a constraint. Constraints allow you to bound certain values for your optimization. On the left side you are allowed to put either an optimization parameter (one of the R, L, or C values you selected in the earlier page) or a voltage node. If you put an optimization parameter <= or >= support only constant numbers on the right side. If you have an optimization parameter (R,L,or C value) and put = you can put mathematical expressions featuring constants as well as additional optimization parameters (R, L, or C values). If you put a voltage node you can only use <= or >=. = Does not work for voltage nodes. In addition the right side can only be a constant value. If you input an invalid constraint the program will error and let you know.



You can also import and export constraints using the buttons below. Imported and exported constraints only work for the same netlist.

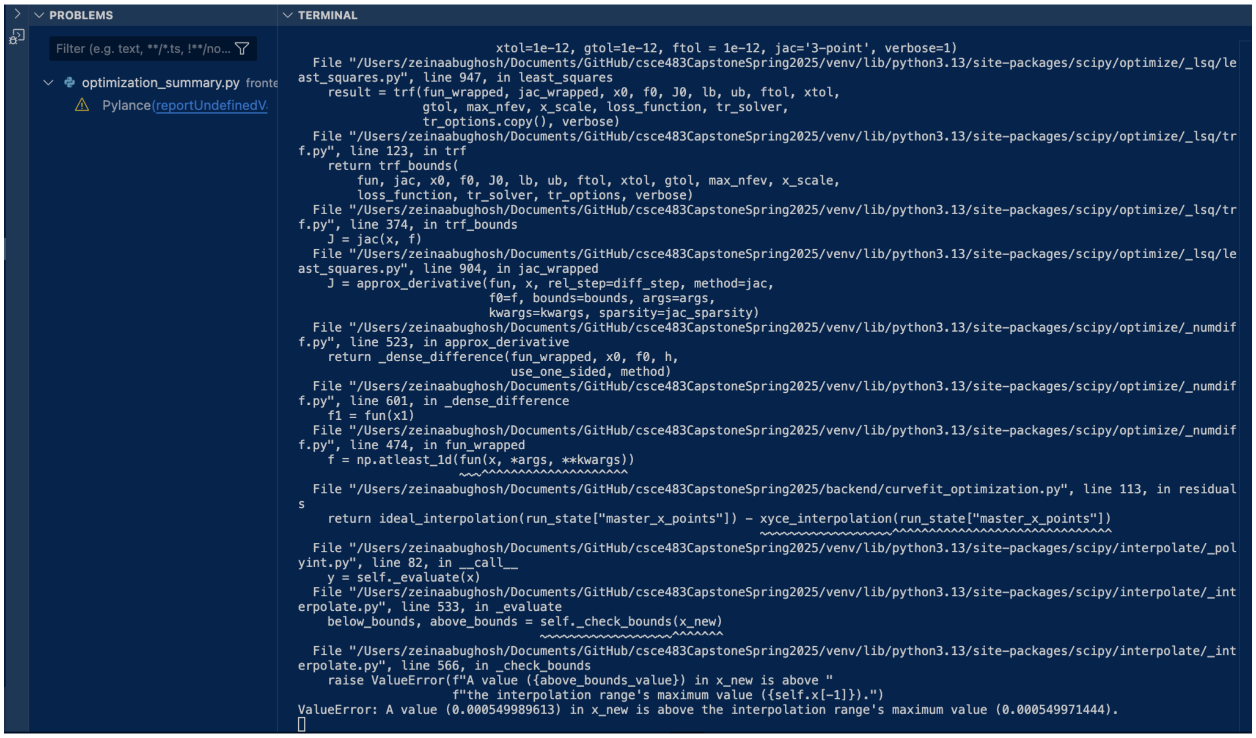


Optimization In Progress

Parameter	Value
Update:	Optimization Results: [40, 8, 51.273, 0.0, 0.0]
Update:	total runs completed: 40
Update:	total runs completed: 35
Update:	total runs completed: 30
Update:	total runs completed: 25
Update:	total runs completed: 20
Update:	total runs completed: 15
Update:	total runs completed: 10
Update:	total runs completed: 5

Afterward, the optimization will be complete. The optimized netlist will be present as a copy in the original directory where the netlist is located, with copy.txt added to the end of the file.

Common Errors


PROBLEMS
Filter (e.g. text, **/*.ts, !**/no...
optimization_summary.py fronte
Pylance(reportUndefinedV
TERMINAL

```
                           xtol=1e-12, gtol=1e-12, ftol = 1e-12, jac='3-point', verbose=1)
  File "/Users/zeinaabughosh/Documents/GitHub/csce483CapstoneSpring2025/venv/lib/python3.13/site-packages/scipy/optimize/_lsq/le
ast_squares.py", line 947, in least_squares
    result = trf(fun_wrapped, jac_wrapped, x0, f0, J0, lb, ub, ftol, xtol,
                 gtol, max_nfev, x_scale, loss_function, tr_solver,
                 tr_options.copy(), verbose)
  File "/Users/zeinaabughosh/Documents/GitHub/csce483CapstoneSpring2025/venv/lib/python3.13/site-packages/scipy/optimize/_lsq/tr
f.py", line 123, in trf
    return trf_bounds(
        fun, jac, x0, f0, J0, lb, ub, ftol, xtol, gtol, max_nfev, x_scale,
        loss_function, tr_solver, tr_options, verbose)
  File "/Users/zeinaabughosh/Documents/GitHub/csce483CapstoneSpring2025/venv/lib/python3.13/site-packages/scipy/optimize/_lsq/tr
f.py", line 374, in trf_bounds
    J = jac(x, f)
  File "/Users/zeinaabughosh/Documents/GitHub/csce483CapstoneSpring2025/venv/lib/python3.13/site-packages/scipy/optimize/_lsq/le
ast_squares.py", line 904, in jac_wrapped
    J = approx_derivative(fun, x, rel_step=diff_step, method=jac,
                          f0=f, bounds=bounds, args=args,
                          kwargs=kwargs, sparsity=jac_sparsity)
  File "/Users/zeinaabughosh/Documents/GitHub/csce483CapstoneSpring2025/venv/lib/python3.13/site-packages/scipy/optimize/_numdif
f.py", line 523, in approx_derivative
    return _dense_difference(fun_wrapped, x0, f0, h,
                             use_one_sided, method)
  File "/Users/zeinaabughosh/Documents/GitHub/csce483CapstoneSpring2025/venv/lib/python3.13/site-packages/scipy/optimize/_numdif
f.py", line 601, in _dense_difference
    f1 = fun(x1)
  File "/Users/zeinaabughosh/Documents/GitHub/csce483CapstoneSpring2025/venv/lib/python3.13/site-packages/scipy/optimize/_numdif
f.py", line 474, in fun_wrapped
    f = np.atleast_1d(fun(x, *args, **kwargs))
                      ~~~^^^^^^^^^^^^^^^^^^^^^
  File "/Users/zeinaabughosh/Documents/GitHub/csce483CapstoneSpring2025/backend/curvefit_optimization.py", line 113, in residual
s
    return ideal_interpolation(run_state["master_x_points"]) - xyce_interpolation(run_state["master_x_points"])
                                                               ~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

  File "/Users/zeinaabughosh/Documents/GitHub/csce483CapstoneSpring2025/venv/lib/python3.13/site-packages/scipy/interpolate/_pol
yint.py", line 82, in __call__
    y = self._evaluate(x)
  File "/Users/zeinaabughosh/Documents/GitHub/csce483CapstoneSpring2025/venv/lib/python3.13/site-packages/scipy/interpolate/_int
erpolate.py", line 533, in _evaluate
    below_bounds, above_bounds = self._check_bounds(x_new)
                                 ~~~~~~~~~~~~~~~~~~^^^^^^^
  File "/Users/zeinaabughosh/Documents/GitHub/csce483CapstoneSpring2025/venv/lib/python3.13/site-packages/scipy/interpolate/_int
erpolate.py", line 566, in _check_bounds
    raise ValueError(f"A value ({above_bounds_value}) in x_new is above "
                     f"the interpolation range's maximum value ({self.x[-1]}).")
ValueError: A value (0.000549989613) in x_new is above the interpolation range's maximum value (0.000549971444).
```

If you get an error in the command prompt and the optimization summary page that talks about how a value is outside the interpolation range, that means that the Xyce run terminated prematurely. Usually the case of this is that the circuit parameters are too unrealistic to be simulated. The fix for this error is to add constraints to the circuit that make sure that each possible iteration of the circuit can be solved. You can see the netlist values that Xyce errored on by viewing the copy of the netlist file.

XycLOps Code Overview

Table of Contents

Document Purpose

This document seeks to give a broad overview of the codebase that makes up the XycLOps application. The intent of this document is to give future developers an idea of what the existing parts of XycLOps do.

Frontend

main.py

This file is the main entry point for the frontend. It creates the AppController and launches the main Tkinter loop.

utils.py

This file contains misc. functions that the frontend utilizes at various points.

app_controller.py

This file details the data stored by the frontend as well as the logical flow between windows of the user interface. The constructor (invoked by main.py) sets default values for important application data and shows the netlist uploader window.

netlist_uploader.py

This file details the first window of the application that allows users to upload their netlist file. It does this by creating a button that utilizes a function detailed in utils.py to open a file system browser. It then creates a button that uses the AppController's navigate function to launch the next window, parameter selection.

parameter_selection.py

This file builds the UI for parameter selection. It does this by creating a Netlist object with the netlist file path as an argument, triggering the **init** function detailed in netlist_parse.py. This information is then displayed in a selectable list, and a button allows navigation to the next window, optimization settings.

optimization_settings/optimization_settings_window.py

This file builds the base UI for the optimization settings screen. It relies heavily on other UI and processing functions contained in the optimization_settings directory for many things to increase clarity since this window is the most complicated. This menu is used to fill a large amount of application data that provides constraints and parameters for the optimization process. A button allows navigation to the final window, optimization summary.

optimization_summary.py

This file builds the UI for the optimization summary screen and then launches a thread that calls the optimization process. The optimization process populates a queue that the frontend can then consume from to continuously update a status report and graph showing optimization progress. The optimization process continues until finished, and a button then allows the user to exit the application.

optimization_settings/add_constraint_dialog.py

See below

optimization_settings/constraint_table.py

See below

optimization_settings/curve_fit_settings.py

See below

optimization_settings/edit_constraint_dialog.py

See below

optimization_settings/expression_evaluator.py

These 5 files are used by the optimization settings window for both utility functions and to build certain UI elements. They are separated into separate files to encapsulate some complexity in the main optimization settings window file.

Backend

curvefit_optimization.py

This file contains the main optimization loop function, curvefit_optimize. This function takes as input a target value (i.e. a particular node voltage), a target curve (list of ideal time vs voltage pairs), a Netlist object with circuit part information, a writable file path to write a new file, and two data structures detailing node and part constraints. It then uses SciPy's least_squares function to find the best combination of part value variations according to many different criteria that match the target input curve. It does this through the repeated computation of a residual by invoking Xyce and comparing how test part values compare and approach the ideal target curve. This file then outputs the optimal values to the writable file path and returns an array with key optimization statistics.

netlist_parse.py

This file contains the class definitions for both Component and Netlist. Component is a simple data structure that saves vital data about individual parts of a circuit. At its core, Netlist is a data structure that represents a condensed netlist. Netlist stores an array of Components, an array of nodes, and a file path to the netlist. It also provides functionality to parse netlist files, write itself out to a netlist file, and add Xyce commands to netlist files.

optimization_process.py

This file contains functions that wrap the main curvefit_optimize function to be invoked by the frontend. It prepares data provided from the front end to be the arguments for the curvefit_optimize function. It then populates a queue with information that can be consumed by the frontend.

xyce_parsing_function.py

This file contains functionality for parsing Xyce process output. Xyce outputs .prn files that can be configured to be formatted in a variety of styles. These functions expect CSV-style file input and convert this data to structures that Python can use (arrays, tuples, etc.).

XycLOps Vision

Table of Contents

Document Purpose

This document seeks to outline the developers' thoughts on future improvements that could be made to XycLOps. These observations are drawn from experience built during tool development and use. Future contributors to the XycLOps tool are encouraged to draw inspiration from the following to make meaningful and impactful contributions.

List

- ### Extraneous File I/O Reduction
  - Currently, every Xyce iteration outputs a file that is in turn parsed into memory for XycLOps's data operations. This means every Xyce iteration has two associated file I/O operations. Since I/O bound operations are notoriously slow, finding a way to eliminate this would most likely improve tool performance. (Maybe something like writing files to /dev/shm or memory mapped files)
- ### Parallelism
  - Currently, all Xyce iterations are run serially. Finding a way to run Xyce iterations in a parallel fashion, perhaps even utilizing GPUsm would certainly increase tool performance. Note: This is hard and could require major optimization loop changes since there does not appear to be a trivial way to parallelize SciPy least_squares.

- ## Multi-platform Support
    - Currently, this tool is tailored to support a Windows experience. While other platforms do work, keeping broad multi-platform support in mind throughout the future development process would improve the tool’s versatility and reach.
- ## Robust User Experience
    - Currently, Xyce offers few user experience features such as integrated explanations of key components and user input checks. Adding such features would make the tool more robust and resistant to unskilled usage.
- ## Non-Transient Analyses
    - Currently, the tool only supports transient analysis using Xyce. Increasing the tool’s scope to support different analysis types that Xyce can use, especially AC analysis, would greatly increase the utility of this tool.
- ## Optimization Settings
    - Currently, the tool uses SciPy’s least_squares function for optimization workload with many key parameters hardcoded. Making these parameters customizable by the user (with proper explanation) would make the tool much more responsive to the diverse use cases that entail different computational rigor for the optimization engine.
- ## Model Compatibility
    - Currently, XycLOps can only work with certain circuit models. Making the tool compatible with common circuit models and libraries (e.g. LTSpice default models) would greatly increase the tool’s reach and give users more freedom in their choice of schematic editor.
- ## Non-curvefit Optimization
    - Currently, XycLOps only accepts specified voltage curve input as an optimization criteria. Adding more types of optimization criteria could increase applicable use cases.
- ## Schematic Editor Integration
    - Currently, XycLOps is only loosely coupled with a schematic editor, relying on only the netlist files that are generated by the editor. More closely integrating XycLOps with a schematic editor, graphically and in terms of error handling, could greatly improve user experience.
- ## Multiple Optimizations
    - Currently, the parameter setup work for XycLOps allows a user to run only one optimization. Changing this to allow multiple optimizations with varying tran commands would allow users to do more with less setup time.

- ## Xyce Error Visibility
    - Currently, XycLOps assumes that user netlists will run perfectly on Xyce. Giving the tool the ability to correct basic Xyce errors such as unit formatting and displaying specific Xyce errors to users would make the tool much easier to use for Xyce novices.