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.).