Plato

Release 0.0

Plato Team

Jul 27, 2021

CONTENTS

Note: These pages are a pre-release version. They are under active development.

These Github pages are meant to serve as the documentation for **Plato Engine** and its associated repositories. These repositories can be found at the Plato Engine GitHub account.

These pages are built using the Plato Docs GitHub repository.

CHAPTER

ONE

CONTENTS

1.1 Description

The **PLATO Engine** computer program serves as a collaborative testbed rich in light-weight synthesis tools for optimization-based design. PLATO Engine is a research code designed to facilitate collaboration with academia, labs and industries by providing interfaces for plug-n-play insertion of synthesis technologies in the areas of modeling, analysis and optimization. Currently, PLATO Engine offers a set of light-weight tools for finite element analysis, linear- and nonlinear-programming, and non-gradient based optimization. The PLATO Engine program is designed to run on high-performance computers.

1.1.1 Application

The PLATO Engine testbed is designed to support research in the area of optimization-based design on high-performance computing systems. The PLATO Engine testbed is used to explore interoperability with several analysis, modeling and optimization tools. The testbed is also used to test the viability of these analysis, modeling and optimization tools for the solution of optimization-based design problems.

1.1.2 Approach

PLATO Engine is intended to serve as a collaborative testbed. It is designed to enable intercommunication of modeling, analysis and optimization data using a Multiple Program, Multiple Data (MPMD) parallel programming model. The MPMD model allows multiple, independent programs/executables to share data in-memory. The optimization algorithms orchestrate the execution and communication between multiple analysis codes and aggregates their contributions to create designs that meet multiple performance criteria.

1.1.3 High Performance Computing

PLATO Engine has been designed for MPMD parallel executions. It also targets Single Program, Multiple Data (SPMD) parallel programming model; however, the SPMD model is not heavily used by the targeted PLATO applications. PLATO Engine is also performance-protable, which allow it to optimally perform in current and next-generation computing architectures.

1.1.4 Required Libraries

Trilinos library (provides Epetra, Seacas and STK): https://github.com/trilinos/trilinos

Omega_h library (provides mesh metadata): https://github.com/SNLComputation/omega_h

Netcdf library (provides I/O libraries): https://www.unidata.ucar.edu/software/netcdf/

AMGX library (provides GPU linear solver): https://github.com/NVIDIA/AMGX

Lapack library (provides linear algebra libraries): http://www.netlib.org/lapack/

Boost library (provides C++ source libraries): https://www.boost.org/

1.1.5 Hardware Requirements

Tested compilers are `g++ 4.7.2`, `g++ 5.4.0`, and `g++ 7.2.0` compilers. Tested OS include Linux and Mac. RAM requirements are problem size dependent.

TODO: is the following still true?

Note: Currently, PLATO Analyze only runs on Graphics Processing Units (GPUs).

1.1.6 Contributing

Please open a GitHub issue to ask a question, report a bug, request features, etc. If you'd like to contribute, please fork the repository and use a feature branch. Make sure to follow the team's coding style policies. Pull requests are welcomed.

TODO: create coding style policies page

1.1.7 User Support

Users are welcomed to submit questions via email to plato3D-help@sandia.gov.

1.2 Plato Repositories

These repositories can be found at the Plato Engine GitHub account.

GitHub Repository	Description
AMGX	Distributed multigrid linear solver library on GPU
exo2obj	Convert from an exodus volume mesh to an obj surface mesh
nvcc_wrapper	Wrapper shell script for NVIDIA nvcc to better conform to host compiler command line arguments
Plato3D	
Plato3D_backend	
platoanalyze	Fast physics and gradients for multidisciplinary analysis and optimization
platodocs	Documentation for Plato
platoengine	Plato Engine - Optimization-Based Design Ecosystem
spack	The spack configuration repository for building Plato.
tests	
use_cases	Collection of Plato Engine use cases
verification	verification and validation problems

1.3 Documentation For Plato

This section describes the process for cloning, editing, and commiting to these GitHub pages, i.e., these web pages. These pages are created using reStructuredText in Sphinx.

1.3.1 Cloning Plato Docs

Navigate to a directory where `/platodocs/` and `/platodocs-build/` can be stored.

```
$ git clone https://github.com/platoengine/platodocs.git

$ mkdir platodocs-build

$ cd platodocs-build

$ git clone https://github.com/platoengine/platodocs.git html

$ cd html

$ git checkout -b gh-pages remotes/origin/gh-pages
```

The directory structure should look something like this:

```
topfolder\
|-- platodocs\              <-- release branch
|   |-- docs\
|   |   |-- source\
|   |   |   |-- images\
|   |   |   |-- _static\
|   |   |   |-- _templates\
|   |   |   |-- conf.py     <-- configuration information to sphinx (extensions, etc.)
|   |   |   |-- index.rst   <-- index source file
|   |   |   `-- other files
```

(continues on next page)

(continued from previous page)

```
|   |   |-- make.bat
|   |   `-- Makefile
|   |-- platodocs.pdf
|   `-- README.md
|
`-- platodocs-build\
    |-- doctrees\
    `-- html\           <-- gh-pages branch
        |-- _images\
        |-- _source\
        |-- _images\
        |-- index.html  <-- main html file (can open with browser)
        |-- .nojekyll   <-- tells github to not use a jekyll theme
        |-- README.md
        `-- other files
```

The existing webpages can be viewed by opening `/platodocs-build/html/index.html` in a web browser or they can be viewed in the pdf version from `/platodocs/platodocs.pdf`.

1.3.2 Editing Plato Docs

Editing Existing Pages:

Documentation can be edited from the source folder (`/platodocs/docs/source/`). The files in this directory are written using reStructuredText (.rst). These files can be opened in any text editor and edited using syntax found here.

Adding New Pages:

When adding a new page to the html web pages, the user can create another .rst file in the source directory (`/platodocs/docs/source/`). This file is added to the website by adding the file name to the contents in `/platodocs/docs/source/index.rst` or another file if the user does not want the file to show up on the home page.

For example, this page is built based on the `documentation.rst` and was added to the web pages by using "toctree" directive in `index.rst`. An example of this is shown below.

`/platodocs/docs/source/index.rst`

```
.. toctree::
:maxdepth: 2

   description
   githubrepositories
   documentation        <-- adding documentation.rst
```

Updating HTML and PDF in Ubuntu:

The web pages (`.html` files) can be updated to represent the changes in the `.rst` files.

First, the user should navigate to `/platodocs/docs/`. Then they can run the following command in a virtual environment (.venv) from the terminal. For directions to get into a virtual environment, go to (**TODO:** add reference to virtual environment).

```
(.venv) $ make html
```

Changes can be viewed by opening `/platodocs-build/html/index.html` in a web browser.

The user can update the pdf version of the documentation by running the following command at from `/platodocs/docs/` from the terminal.

```
(.venv) $ make latexpdf
```

The changes can be viewed by opening `/platodocs/platodocs.pdf`.

1.3.3 Publishing Documentation

In order to publish changes to the Plato Docs GitHub Pages the user will have to Git Push changes to the build repository (`gh-pages` branch). It is also wise to Git Push the changes in the source files (`release` branch). See instruction on *pushing and pulling* to both branches of Plato Docs.

1.3.4 Pushing and Pulling

The process for pushing to and pulling from the `release` branch of Plato Docs (`/platodocs/`) is standard. This is the branch used for the source files of Plato Docs.

In `/platodocs/` :

```
$ git push
```

or

```
$ git pull
```

The user will have to specify the branch when pushing to or pulling from the `gh-pages` branch of Plato Docs (`/platodocs-build/`). This is the branch used for the build files of Plato Docs.

In `/platodocs-build/` :

```
$ git push origin gh-pages
```

or

```
$ git pull origin gh-pages
```

1.3.5 Installing Sphinx and Other Packages

The sphinx documentation is set up to work with Python3 packages. The user will need to install pip, virtualenv, and Sphinx.

Installing Some Packages

The following code block installs some necessary packages that are probably already installed if the user has installed Plato.

```
$ sudo apt-get update
$ sudo apt-get -y upgrade
$ sudo apt-get -y install build-essential curl git gfortran python python-dev vim tcl␣
↪environment-modules unzip csh python3-distutils
```

Installing pip

Sphinx packages are published on the Python Package Index. The preferred tool for installing packages from *PyPI* is **pip**. Ubuntu users have to manually install **pip** for Python3.

```
$ sudo apt install python3-venv python3-pip
```

Installing virtualenv

To install the virtual environment to manage Python packages:

```
$ python3 -m pip install --user virtualenv
```

Installing Latex Packages

The user will have to install certain latex packages to enable building a pdf version of Plato Docs.

```
$ sudo apt-get install texlive-latex-base texlive-fonts-recommended texlive-fonts-extra␣
↪texlive-latex-extra latexmk
```

Installing Sphinx

Finally, the user will have to install Sphinx from a virtual environment in order to get the most recent version:

```
$ python -m venv ~/.venv
$ source ~/.venv/bin/activate
(.venv) $ python -m pip install sphinx
```

Check that the Spinx version is equivalent or beyond `sphinx-build 4.0.0`

```
(.venv) $ sphinx-build --version
```

1.3.6 Virtual Environment

Sphinx build commands (`$ make html` or `$ make latexpdf`) will have to done from a Python3 virtual environment. To create a virtual environment:

```
$ python3 -m venv ~/.venv
```

Note that the virtual environment only has to be created once.

Once the virtual environment is created it can be sourced:

```
$ source ~/.venv/bin/activate
```

Check that default Python version is 3.0.0 or later and Sphinx is 4.0.0 or later:

```
(.venv) $ python --version
(.venv) $ sphinx-build --version
```

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search