OptiPuls

Dmytro Strelnikov, Roland Herzog

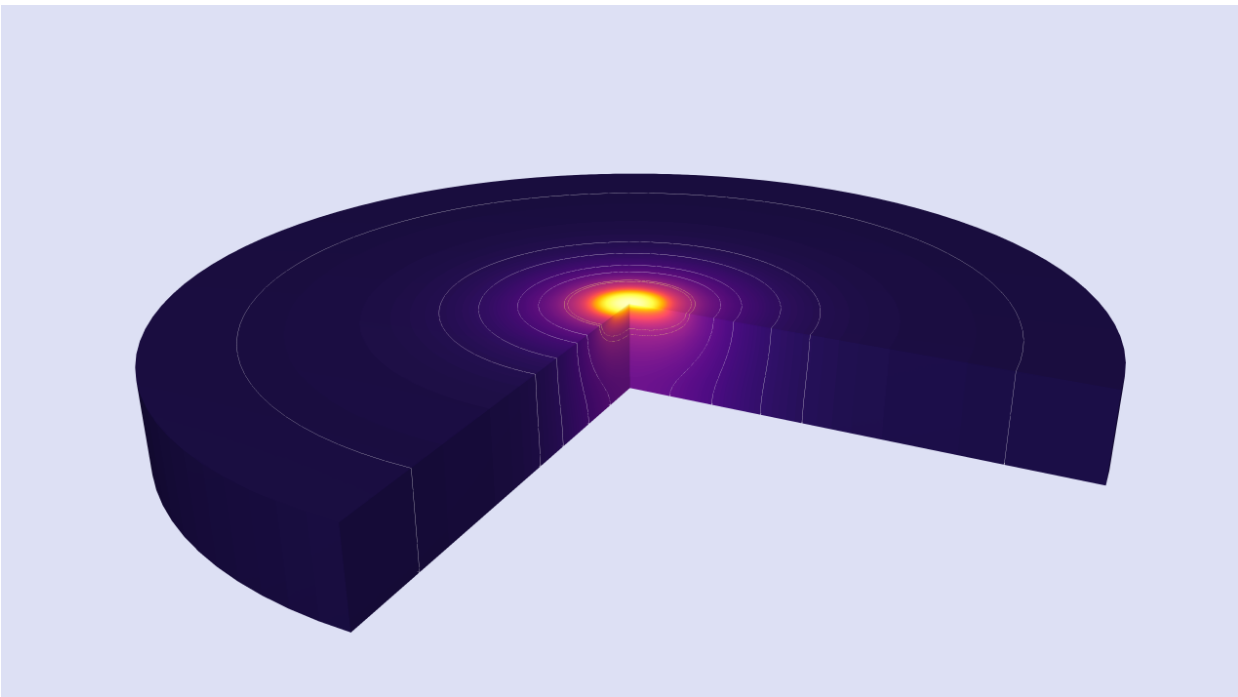
Oct 28, 2022

CONTENTS:

CHAPTER

ONE

PROJECT INFO



Authors: Dmytro Strelnikov, Roland Herzog

Funding: Funding: Project IGF 20.826B (DVS I2.3005) in Forschungsvereinigung Schweißen und verwandte Verfahren e.V. of the Deutschen Verbandes für Schweißen und verwandte Verfahren e.V.

Project page: Simulation Based Optimization of the Time-Dependent Pulse Power for Laser Beam Welding of Aluminum Alloys in Order to Avoid Hot Cracks

This documentations is dedicated to the python package *optipuls* which implements a numerical model for simulation and mathematical optimization of the single spot pulsed laser beam welding of aluminium alloys. Its implementation strongly relies on the free and open-source FEniCS computing platform.

This software package allows an expert user to create numerical simulations for the single-spot pulsed laser beam welding process, formulate in a flexible way and solve the corresponding optimal control problems, and finally generate optimized laser pulse shapes. These pulse shapes can be further programmed into a laser's power source in order to obtain fast, energy efficient, and crack-free welds.

1.1 Useful Links

- https://github.com/optipulsproject/python-optipuls
- https://github.com/optipulsproject/optimal-control-spot-welding
- https://hub.docker.com/r/optipulsproject/optipuls

CHAPTER

TWO

INSTALLING `OPTIPULS`

Note: The `optipuls` package is not a click-to-run software. While this documentation covers the main points for a quick start, it remains an expert oriented system. Some knowledge of the linux command line, docker, and python programming language is required.

2.1 Requirements

To run simulations and solve optimization problems, `optipuls` requires a working FEniCS installation. Python packages `numpy`, `scipy` and `matplotlib` must be already installed as FEniCS' dependencies.

Notice, that installing FEniCS with its dependencies might be difficult on systems other than Debian or Ubuntu, which are the ones [officially supported][fenics/installation/ubuntu] by FEniCS developers. Therefore, it is recommended to use `optipuls` in a docker container using [optipulsproject/optipuls] docker image which is built on top of [fenicsproject/stable] docker image. Please refer to the section below.

2.2 Installing `optipuls` on the host system

Provided FEniCS is correctly deployed on the host system, `optipuls` can be simply installed via `pip`.

Creating a Python virtual environment and switching to it (optional):

```
python3 -m venv optipulsenv
source optipulsenv/bin/activate
```

Installing `optipuls`:

```
python3 -m pip install git+https://github.com/optipulsproject/optipuls
```

2.3 Running `optipuls` in a docker container

The `optipuls` python package is based on the free and open-source FEniCS computing platform for solving partial differential equations (PDEs). The FEniCS project ships with its official docker images.

In view of this, the `optipuls` python package comes in a bundle with the official optipuls docker images available at https://hub.docker.com/r/optipulsproject/.

Getting the OptiPuls docker image on a system with docker software installed is as simple as:

```
$ docker pull optipulsproject/optipuls:latest
latest: Pulling from optipulsproject/optipuls
Digest: sha256:89015703048a0ad76d0a11880f763e20bbe8cb8903db977b785b702c432e22df
Status: Downloaded newer image for optipulsproject/optipuls:latest
docker.io/optipulsproject/optipuls:latest
```

2.3.1 Useful Links

- Docker Overview
- Getting Started with Docker
- Install Docker Engine

2.4 Extras

ParaView is recommended to inspect the simulation output and Gmsh is needed to view `.MSH` files.

CHAPTER

THREE

QUICK START

The easiest way to start with `optipuls` is by running and modifying some of the built-in examples.

3.1 Preliminaries

Note: This guide assumes that a Linux system is used. Docker software can be also run on Windows, however this use case was not tested and therefore is not recommended.

It is recommended to create a separate directory on the host system for the output of numerical simulations and optimizations. In this guide it will be `/data/OptiPuls/output`.

```
$ mkdir -p /data/OptiPuls/output # creating a directory for the numerical artifacts
```

The following command will run `optipuls` docker container in interactive mode:

```
$ mkdir /tmp/scratch
$ docker run -it \
        --volume $(pwd):/home/fenics/shared \
        --volume /tmp/scratch:/scratch \
        --user $UID \
        optipulsproject/optipuls:latest

fenics@fc203ef7c5b2:~/shared$
```

The changed command line prompt indicates that the commands will be now run inside the newly created container. To detach and stop the container type `exit` or press *Ctrl-D*.

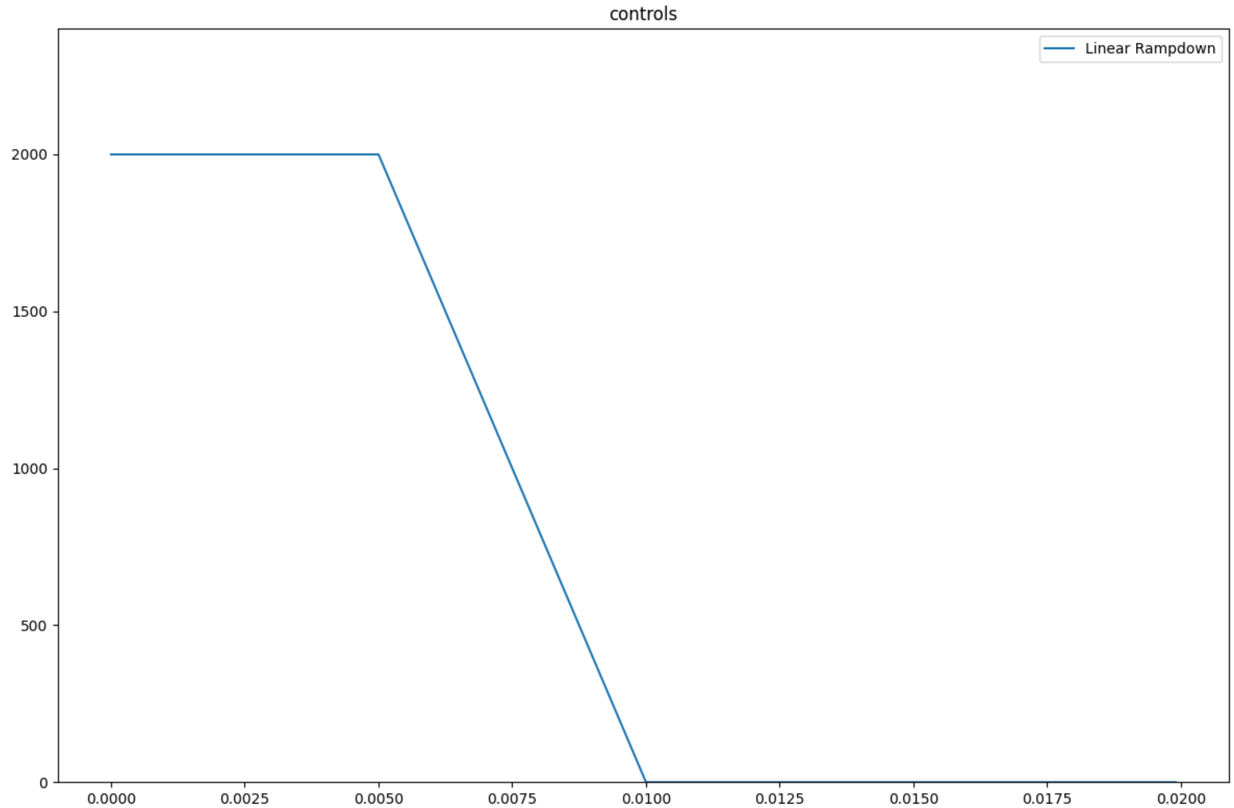
3.2 Linear Rampdown Simulation

Perform the following steps to run a simple built-in numerical simulation for a linear rampdown laser pulse shape.

Inside the container open the directory `python-optipuls` source code and run `examples/simulation_linear_rampdown.py`:

```
fenics@fc203ef7c5b2:~/shared$ cd /home/fenics/src/python-optipuls/
fenics@fc203ef7c5b2:~/shared$ python3 examples/simulation_linear_rampdown.py --scratch /
↪scratch/simulation_linear_rampdown/
```

The resulting files will be in `/data/OptiPuls/output`. Use ParaView (see paraview-helpers.) to inspect the simulation.



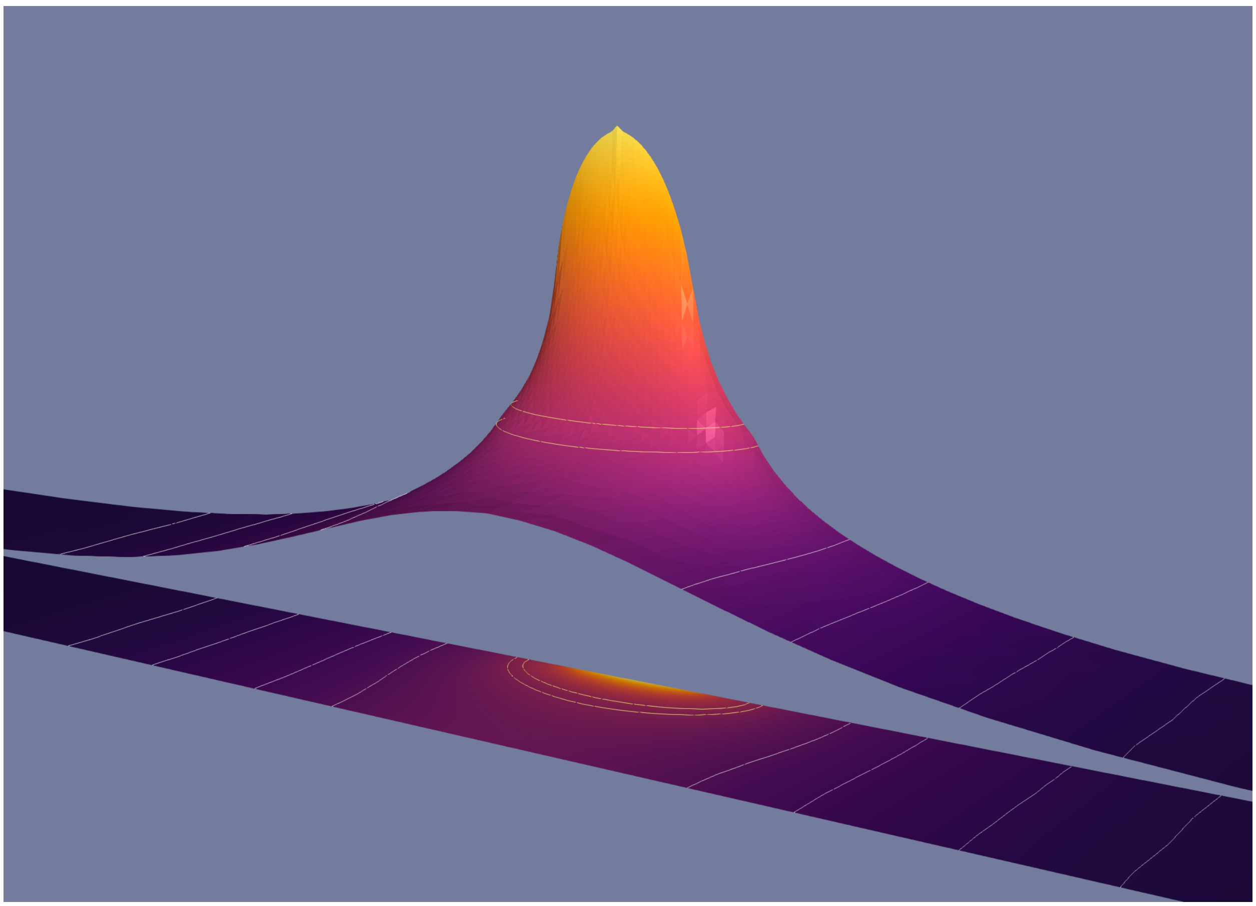
In order to modify this simple example make a copy of the file `simulation_linear_rampdown.py` and follow the comments in the file to adjust the problem formulation. For example, to change the total pulse duration and the rampdown pulse shape one needs to modify the following lines:

```
...
# initialize the time domain
time_domain = TimeDomain(0.020, 200)
...
# create a simulation with linear rampdown pulse shape
control = linear_rampdown(time_domain.timeline, t1=0.005, t2=0.010)
...
```

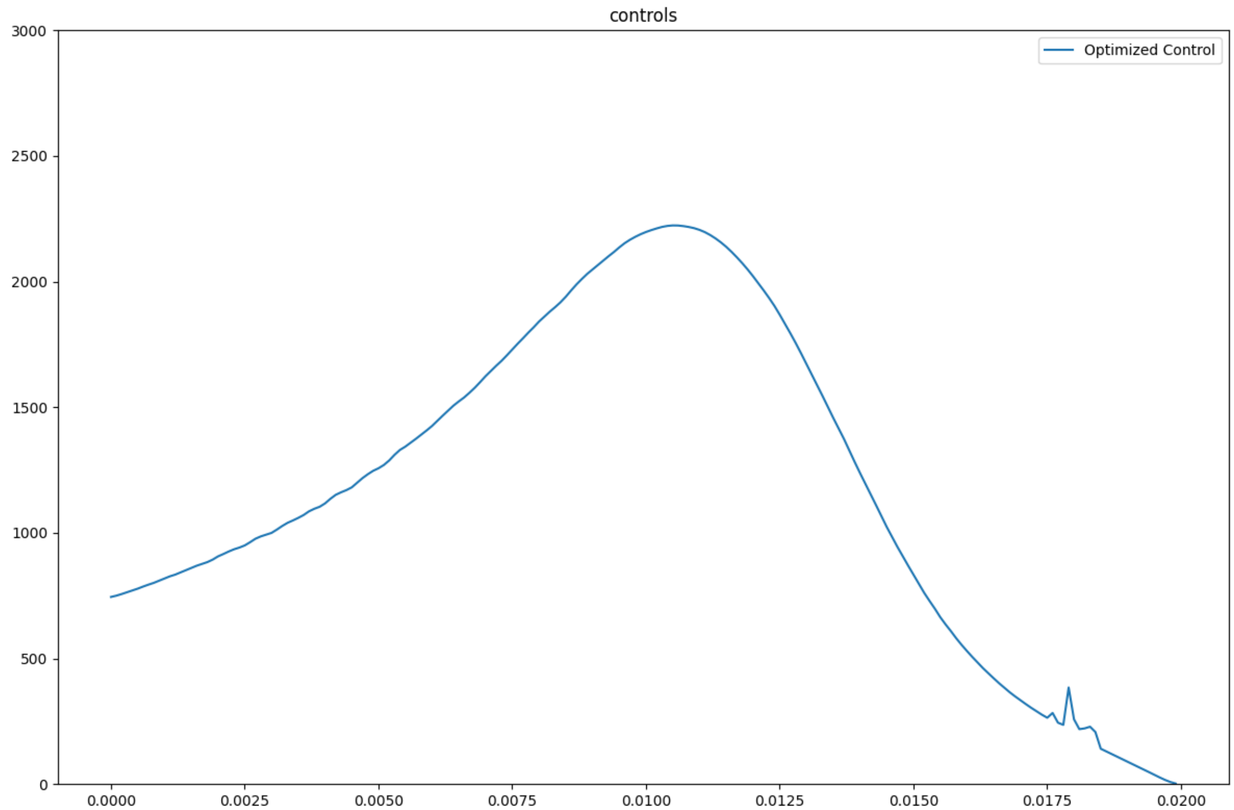
3.3 Zeroguess Optimization

In a similar way toy the previous example, perform the following steps to run a simple built-in numerical optimization with zero pulse as the initial guess. The optimizer will be increasing the laser power untill the desired depth of the weld will be achieved.

In a similar way toy the previous example, start the container, open the directory `python-optipuls` source code and run `examples/optimization_zeroguess.py`:



```
fenics@fc203ef7c5b2:~/shared$ cd /home/fenics/src/python-optipuls/
fenics@fc203ef7c5b2:~/shared$ python3 examples/optimization_zeroguess --scratch /scratch/
↪optimization_zeroguess/
```



One can play around and modify the optimization parameters:

```
# optimization parameters
problem.beta_control = 10**2
problem.beta_velocity = 10**18
problem.velocity_max = 0.15
problem.beta_liquidity = 10**12
problem.beta_welding = 10**-2
problem.threshold_temp = 1000.
problem.target_point = dolfin.Point(0, 0, -.7 * space_domain.Z)
problem.pow_ = 20
```

The impact of these parameters on the produced optimized pulse shapes is explained in the paper *An Optimal Control Problem for Single Spot Laser Pulse Welding*.

CHAPTER

FOUR

AN OPTIMAL CONTROL PROBLEM FOR SINGLE SPOT LASER PULSE WELDING

Repository: https://github.com/optipulsproject/optimal-control-spot-welding
arXiv: https://arxiv.org/abs/2109.10788v2

4.1 Abstract

We consider an optimal control problem for a single-spot pulsed laser welding problem. The distribution of thermal energy is described by a quasilinear heat equation. Our emphasis is on materials which tend to suffer from hot cracking when welded, such as aluminum alloys. A simple precursor for the occurrence of hot cracks is the velocity of the solidification front. We therefore formulate an optimal control problem whose objective contains a term which penalizes excessive solidification velocities. The control function to be optimized is the laser power over time, subject to pointwise lower and upper bounds. We describe the finite element discretization of the problem and a projected gradient scheme for its solution. Numerical experiments for material data representing the EN~AW~6082-T6 aluminum alloy exhibit interesting laser pulse patterns which perform significantly better than standard ramp-down patterns.

4.2 Reproducable numerical results

Note: This paper is witten in a fully reproducible way, i.e. all the numerical artifacts used in the paper are being created every time the paper is being built. While the paper explains the mathematical model behind the core, its source code can be also used as a tutorial for running simulations and optimizations with `optipuls`.

The numerical results presented in the paper can be easily reproduced using following the instructions. These results are based on the corresponding numerical model optipuls.

4.2.1 Why reproducing the result?

We believe that any numerical results presented in a scientific publication must be considered reliable only if the exat way they were obtained is clear and hence they can be verified by a reader. The most transparent way to go is to provide an explicit instruction on reproducing of the results, requiring only free software.

Despite it is often not the case in many scientific publications, we intend to encourage reproducibility culture in computational science by setting an example.

4.2.2 Reproducing (local host system)

A working FEniCS computing platform installation is required as well as the following additional python packages (including their dependencies):

- optipuls
- matplotlib
- tabulate

We suppose that make is already installed on your machine provided a UNIX-like system is used.

If you already have FEniCS installed locally, you can use python virtual environments to install the remaining dependencies without cluttering your system:

```
python3 -m venv --system-site-packages ~/.local/optipuls
source ~/.local/optipuls/bin/activate
pip install git+https://github.com/optipulsproject/optipuls
pip install matplotlib tabulate
```

Once the depencdencies are satisfied, reproducing of the results is as simple as running `make` in the root of the project:

```
git clone https://github.com/optipulsproject/optimal-control-spot-welding
cd optimal-control-spot-welding
make -j$(nproc)
```

Make will run the computations, produce the plots, the tables, and the final `manuscript-numapde-preprint.pdf` file.

4.2.3 Reproducing (docker)

Prebuilt optipuilsproject docker images can be used to reproduce the results provided docker is installed on your system.

Pull neccessary images:

```
docker pull optipulsproject/optipuls:optimal-control-spot-welding
docker pull optipulsproject/tabulate:latest
docker pull optipulsproject/publications:latest
```

Make plots (entails making of the numerical artifacts):

```
docker run \
  -v $(pwd):/home/fenics/shared \
  optipulsproject/optipuls:optimal-control-spot-welding \
  make plots.all -j$(nproc)
```

Make tables:

```
docker run \
  -u $UID \
  -v $(pwd):/data \
  optipulsproject/tabulate:latest \
  make tables.all
```

Make paper:

```
docker run \
  -u $UID \
  -v $(pwd):/data \
  optipulsproject/publications:latest \
  make preprint
```

CHAPTER

FIVE

EXTRAS

5.1 Advanced Mesh Generator

A helper script is provided for generation of advanced problem specific 2d (XZ) and 3d meshes for both the single-spot and the double-spot (multi-spot) problems. Notice that while Gmsh (an open source 3D finite element mesh generator) and pygmsh python package are not required for running optipuls these have to be installed for generation of an advanced custom mesh.

5.1.1 Generating mesh

Listing 1: Mesh generation options.

```
$ python3 mesh_generate.py --help
usage: mesh_generate.py [-h] [-Z Z] [-R R] [-r R] [--overlap OVERLAP] [--dim {2,3}] [--
↪lcar_min LCAR_MIN]
                        [--lcar_max LCAR_MAX] [-o OUTPUT] [-v] [--singlespot]

options:
  -h, --help            show this help message and exit
  -Z Z                  height of the problem domain
  -R R                  radius of the problem domain
  -r R                  radius of the laser beam
  --overlap OVERLAP     overlap of the welding spots for double-spot problem, float in␣
↪[0, 1]
  --dim {2,3}           dimension of the mesh
  --lcar_min LCAR_MIN   minimal resolution of the mesh
  --lcar_max LCAR_MAX   maxinal resolution of the mesh
  -o OUTPUT, --output OUTPUT
  -v, --verbose
  --singlespot          use this option for single-spot problems (sets overlap to 1)
```

Examples:

```
$ python3 mesh_generate.py --singlespot --dim 3 --output singlespot_XYZ.msh
$ python3 mesh_generate.py --overlap=.5 --dim 3 --output doublespot_0.5_XYZ.msh
$ python3 mesh_generate.py --singlespot --dim 2 --output singlespot_XZ.msh
```

The generated files can be viewed in Gmsh.

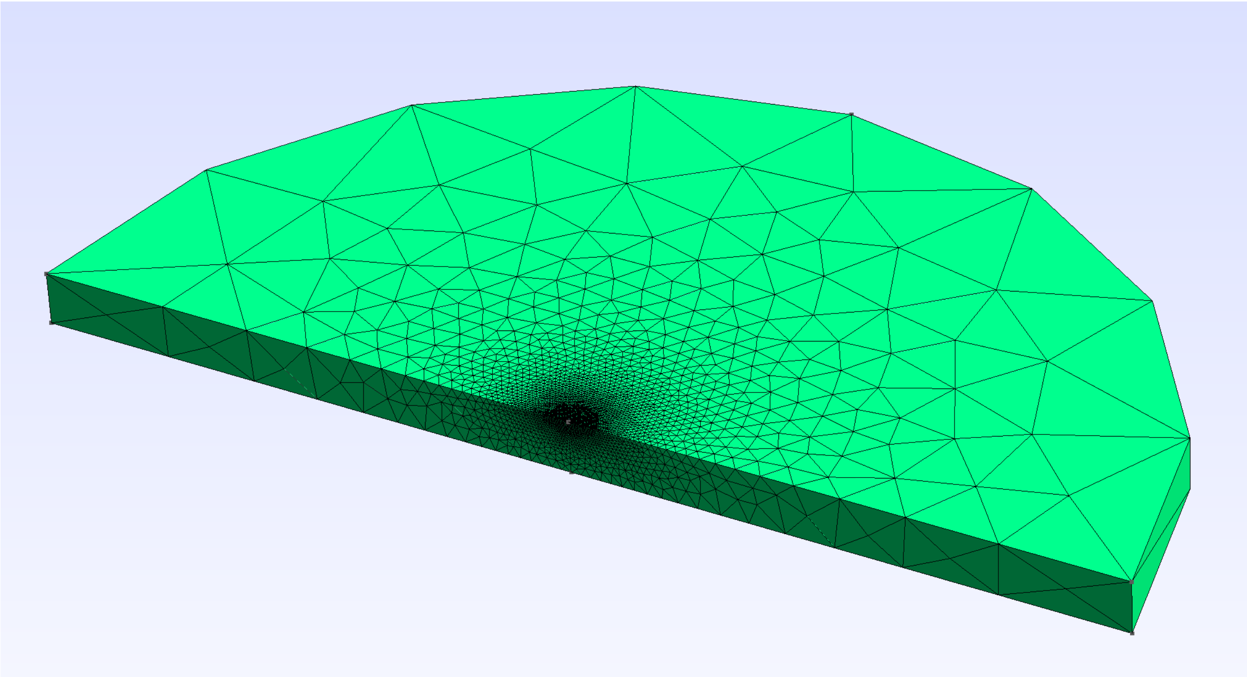


Fig. 1: Inspect generated mesh in Gmsh.a

5.1.2 Converting mesh

In order to be used by FEniCS the mesh should be converted to `.XDMF` format.

Listing 2: Mesh convertion options.

```
$ python3 mesh_convert.py --help
usage: mesh_convert.py [-h] [-i INPUT] [-o OUTPUT] [--dim {2,3}]

options:
  -h, --help            show this help message and exit
  -i INPUT, --input INPUT
  -o OUTPUT, --output OUTPUT
  --dim {2,3}           dimension of the mesh
```

Examples:

```
$ python3 mesh_convert.py --dim 3 --output singlespot_XYZ.xdmf
$ python3 mesh_convert.py --dim 2 --output singlespot_XZ.xdmf
```

5.2 ParaView Helpers

ParaView is an open-source, multi-platform data analysis and visualization application. In OptiPuls it is used to inspect the output of the numerical simulation of the laser welding.

5.2.1 ParaView State Files

In order to make the visualization more convenient, a set of preconfigured ParaView state files is provided.

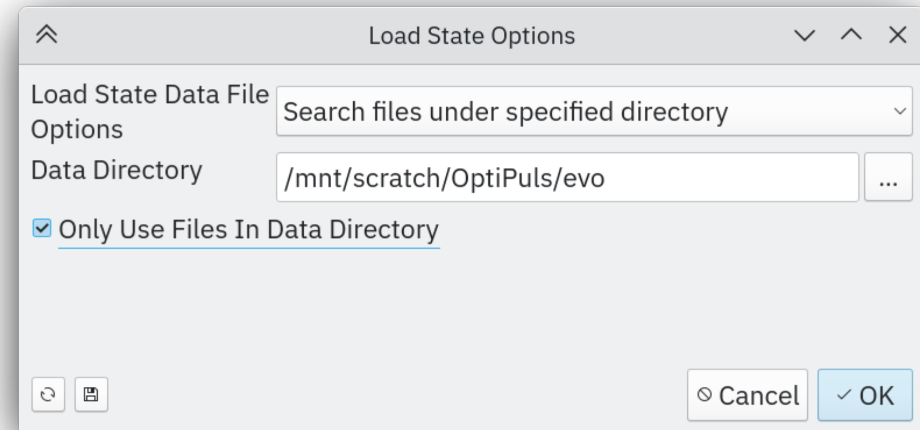


Fig. 2: Load state file and specify the simulation output directory.

5.2.2 Render Animation

A helper script `paraview_save_animation.py` is provided in order to generate an animation for a given state file. Its output is a set of `.PNG` files. These files can be converted to a video using `ffmpeg`.

Example:

```
ffmpeg -r 60 -f image2 -s 3840x2160 -i /tmp/paraview/ani.%04d.png -vcodec libx264 -pix_
↪fmt yuv420p -crf 17 output.mp4
```

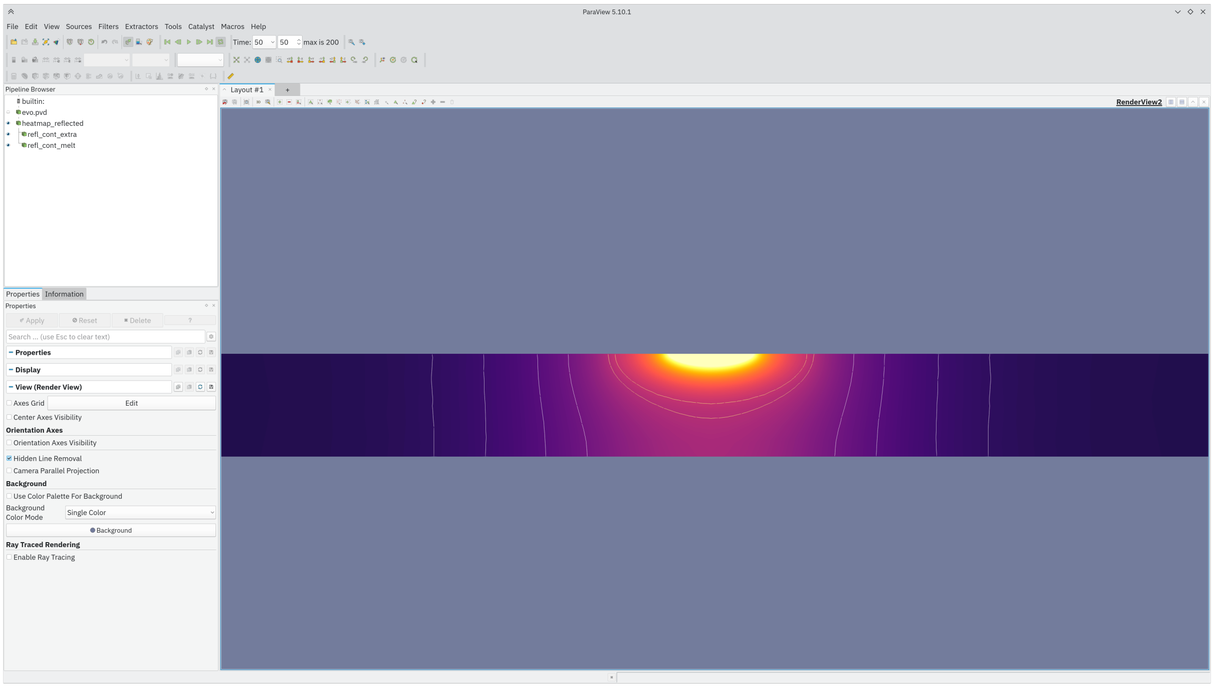


Fig. 3: Sectional view for a 2d problem.

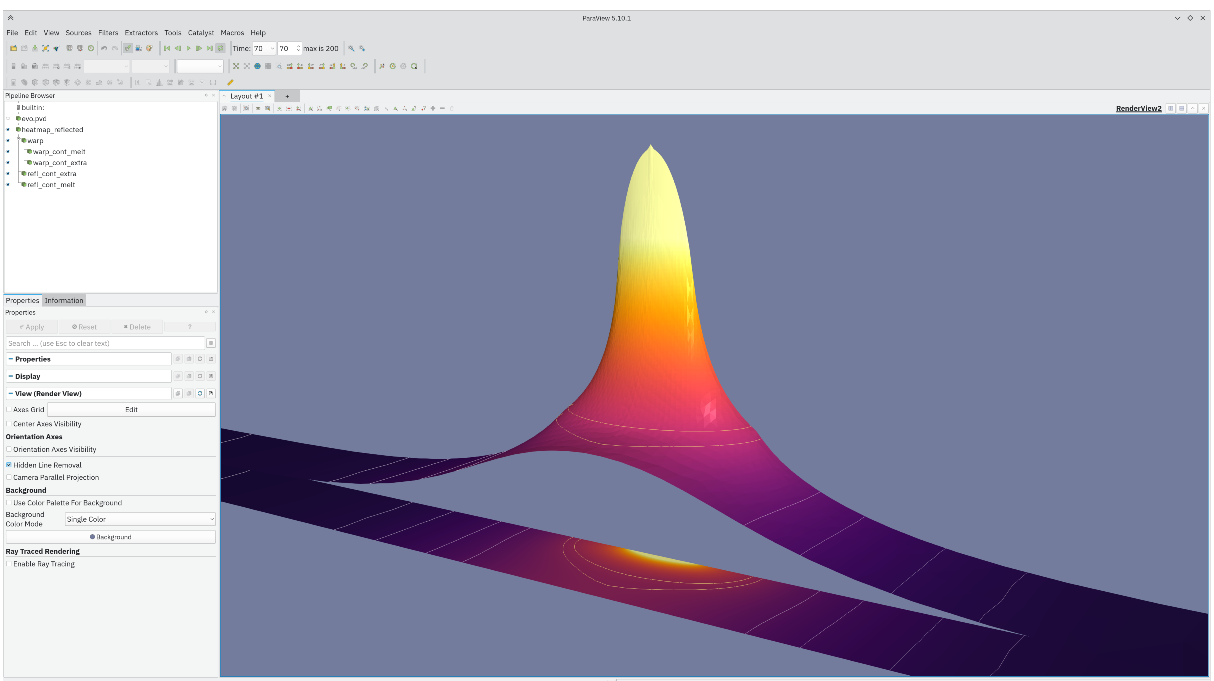


Fig. 4: Warp view for a 2d problem.

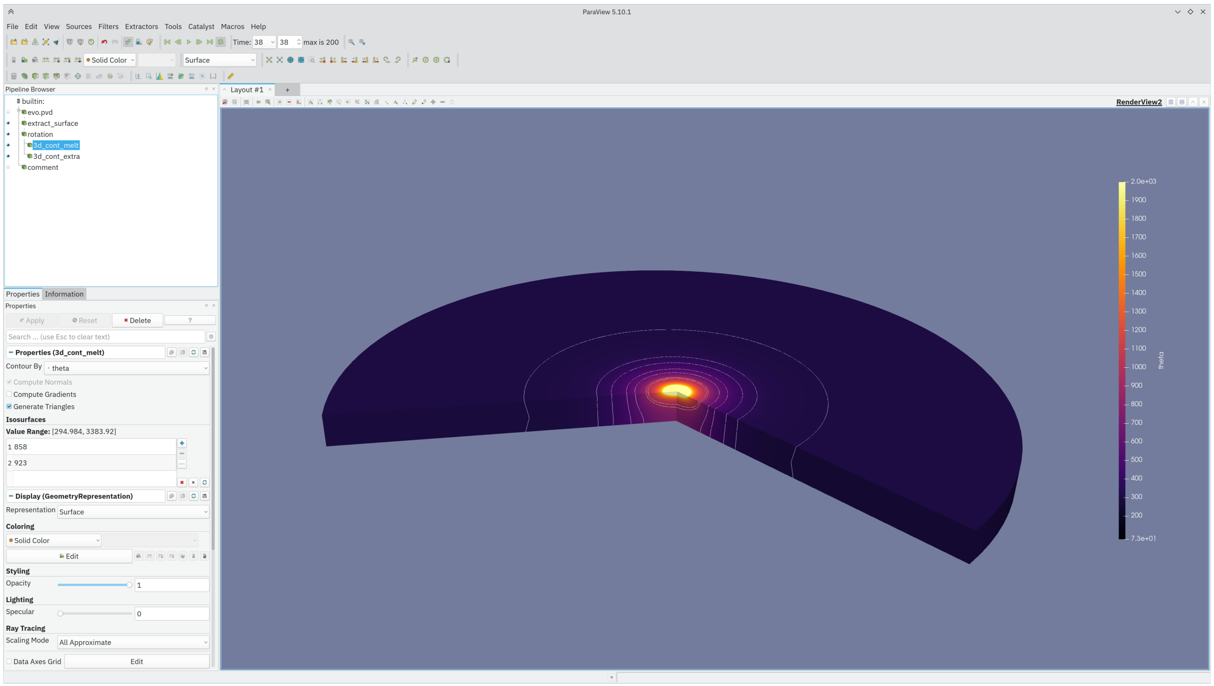


Fig. 5: 3d (rotated) view for a 2d problem.

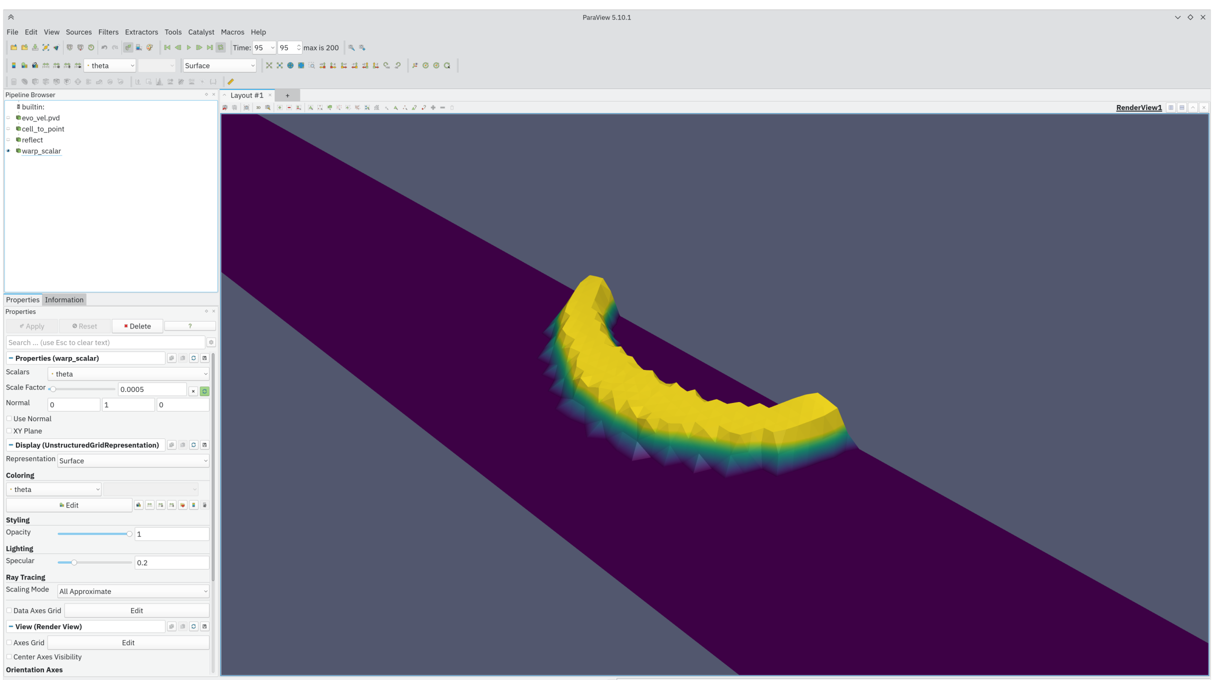


Fig. 6: Solidification front velocity warp view for a 2d problem.

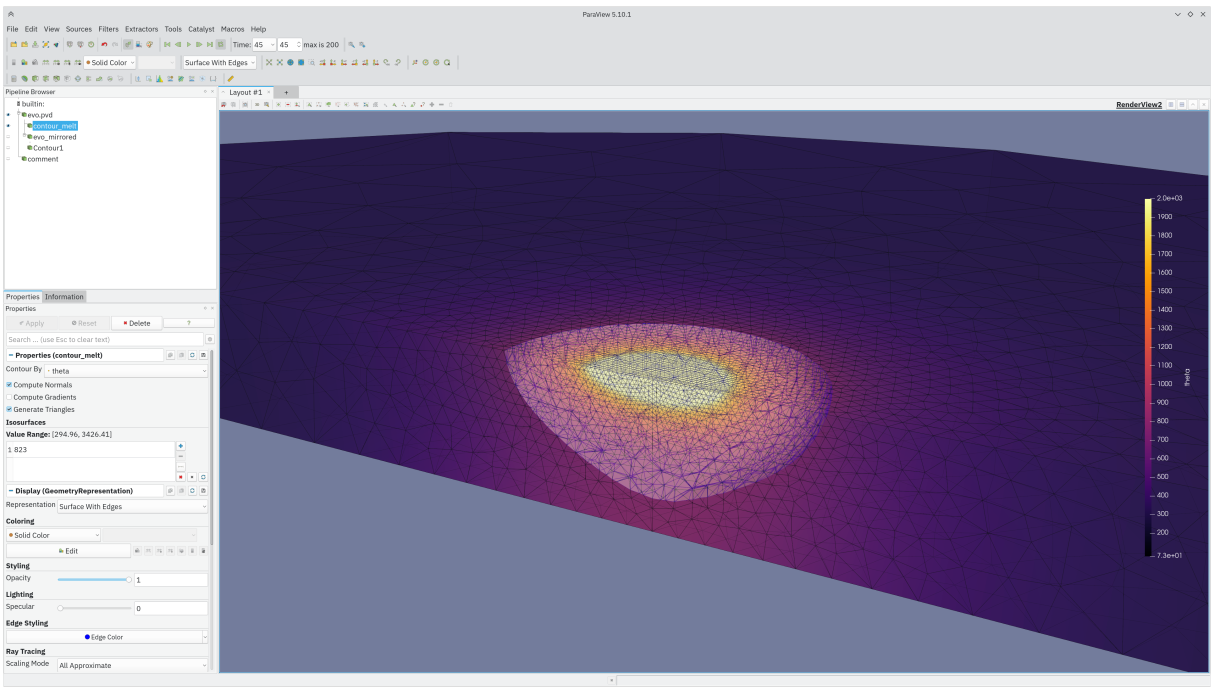


Fig. 7: Wireframe view for a true 3d problem.