UKAEA-CCFE-PR(21)72

J. Shimwell, J. Billingsley, R Delaporte-Mathurin, D. Morbey, M. Bluteau, P.Shriwise

The Paramak, automated parametric geometry construction for fusion reactor designs

The Paramak, automated parametric geometry construction for fusion reactor designs

J. Shimwell, J. Billingsley, R Delaporte-Mathurin, D. Morbey, M. Bluteau, P.Shriwise

This is a preprint of a paper submitted for publication in Fusion Engineering and Design

The Paramak, automated parametric geometry construction for fusion reactor designs

J. Shimwell[1], J. Billingsley[1], R Delaporte-Mathurin[2,3] D. Morbey[4], M. Bluteau[1], P. Shriwise[5], A. Davis[1]

[1] *Culham Centre for Fusion Energy (CCFE), Culham Science Centre, Abingdon, Oxfordshire, OX14 3DB, UK*

[2] *CEA, IRFM, F-13108 Saint-Paul-lez-Durance, France*

[3] *Université Sorbonne Paris Nord, Laboratoire des Sciences des Procédés et des Matériaux, LSPM, CNRS, UPR 3407, F-93430, Villetaneuse, France*

[4] *College of Engineering, Swansea University, Bay Campus, Swansea, SA1 8EN, UK*

[5] *Argonne National Laboratory, 9700 S. Cass Ave., IL 60439, USA*

Abstract

During the design process of fusion reactors it is desirable to rapidly prototype different design concepts and assess their suitability against a range of high level requirements. This helps to narrow the design window and scope out potential designs which can undergo further more detailed analysis. The Paramak is an open-source tool that aims to provide automated parameter driven 3D CAD models for fusion reactor components and magnetic fusion reactors. The geometry produced is compatible with analysis workflows and this allows iterative automated model building and analysis to help steer the design optimisation process. The Paramak uses CadQuery and OCC to create the 3D CAD model. In this paper we demonstrate the use of the Paramak framework to create a few example reactor configurations including: a spherical reactor, a regular large radius tokamak and a compact submersion tank reactor. The models are not exact reproductions of any particular design but just reflective of different reactor designs that are available. Input parameters for the various reactors that the Paramak can generate generally fall into three categories. Which are continuous ranges such as blanket thickness, integer ranges such as number of toroidal field coils and categorical parameters such as type of divertor. The design tool facilitates parameter studies where users can investigate the impact of input design parameters on the reactor performance. The generation of output metrics from input parameters leads itself to the use of data science and machine learning approaches to steer the design.

Keywords: parametric, design, fusion, automated, energy, CAD

1. Introduction

When assessing the suitability of a fusion reactor design one of the stages is the construction of a 3D model. This tends to be a digital 3D CAD model which is then adapted for use in different analysis disciplines, for example, engineering and neutronics. After carrying out the analysis, feedback can be provided and the design cycle can be iterated to refine and optimise the design. Automating the analysis can help to rapidly develop a design. While some automated analysis remains a challenge in certain disciplines, it is generally accepted that automation is progressing in all the required disciplines for analysing fusion reactor designs. The production and adaptation of the 3D CAD models can also be automated to some extent and 3D CAD production is already automated in more mature design disciplines such as aerospace [1]. Typically fusion reactor design processes involve analysis being carried out and fed back into the creation of the next CAD model; this process is mainly a manual GUI based operation. The results from analysis do not directly impact the model generation. Additionally, the situation can occur where the models are updated several times as different analysis streams feedback into the design. This can result in separate analysis streams having different models as inputs. Having an automated model creation for simple space reserving is perhaps the first stage in creating a more automated rapid design cycle. Automated model creation can reduce the risk of geometry creation becoming a bottleneck in the design cycle. While complex model construction might be difficult to automate with the current software, there is perhaps utility in automating simpler models and allowing the analysis specific geometry details to be filled in at a later stage. Additionally there is perhaps also some utility in the use of automated CAD connected with automated analysis at early stages in the design, where simple models are more appropriate.

CadQuery [2] offers a potential solution to the creation of automated parametric CAD. CadQuery is an open-source Python library that binds to OpenCascade [3] and has some unique features among the possible open-source candidates. One such capability is the ability to search, filter and then operate on the CAD solids during construction. This allows components to be linked and built from each other without the operator having to be concerned with redefining related solids when a linked solid is modified. This is already possible with proprietary CAD software but it is now emerging into the open-source area. The combination of permissive licensing and parametric studies allows automated geometry creation and analysis to be carried out on potentially tens of thousands of designs in parallel, using cloud computing without incurring prohibitive costs.

A key advantage of creating a 3D reactor geometry from parameters is that the produced model then becomes easy to quantify in terms of values. Being able to describe a 3D model with a series of parameters allows direct linking between an optimiser, input parameters and output metrics. The designer's input is still required to make the parametrisation rules that allow components to be varied in ways that impact their performance. A designer's skill is required to ensure the layout of components interface correctly and do not overlap. A benefit of the parametric model construction process is that when one parametric model is made this results in many perturbations that can be generated by scripts, while a static model remains a single static model. A disadvantage is that creating a parameterised model layout is more complex than a single model.

2. Software practices employed

The Paramak is an open-source code designed to assist with rapid concept design exploration for fusion reactors. The source-code is under version control and openly available via Github [4] under a permissive MIT licence. The Paramak Python package is distributed via PyPi [5] and there are plans to incorporate a Conda distribution in the future. A containerised build environment is distributed via Dockerhub [6] containing a pre-built environment with all the required dependencies. The code is documented with diagrams and examples on ReadTheDocs [7] which makes use of extensive Docstrings within the code. The code has been professionally reviewed by [8] and has continuous integration and test suite via CircleCI [9]. Github Actions have been utilised from an early stage for automating several aspects of the code distribution, packaging and static code analysis. Github Actions have been used for employing code style guides (PEP8), updating the PyPi package distribution and automatically building and uploading new Dockerhub images with each new version of the code. The decision to open-source the Paramak code was a key enabler which allowed use of the previously mentioned platforms which in turn allowed the code to grow and improve rapidly. Additionally the open-source nature of the project has facilitated contributions from outside the organisation.

3. Code Structure

The Paramak package wraps CadQuery to enable the creation of magnetic confinement fusion reactor models. These reactors often have repeating cross sections around the Z axis and this eases the complexity of simple reactor construction. The Paramak consists of the three main groups of classes: Shapes, Components and Reactors. The Parametric Shapes class provide profiles from a combination of straight edges, circular edges and Bezier spline edges. These shapes can represent a wide range of basic shapes and are made from a series of 2D coordinates. Shapes can be operated on to create 3D volumes using extrude, revolve, sweep and rotate operations. Boolean operations such as cut, intersect and union are also available to Shapes. To build Shapes the class must be provided with coordinates or points to connect up and edge connection types to connect each coordinate.

The Parametric Components inherit from Shape and build upon these basic families of shapes to create volumes that more closely resemble components found in fusion reactors. Parametric components generally have a particular method of finding the coordinates that make up the shape and are thus able to provide the coordinates needed to make a Shape class. The methods of finding points differ from component to component and are encoded within the component's class.

For the simplest parametric components such as CenterColumnCylinder() the points coordinates are found based on a hollow cylinder. This requires just four points and uses straight lines to connect the points followed by a rotation around the Z axis. This is then abstracted for the user so that only the height, inner radius and outer radius are required. The Component class then finds the points from the internal rules and applies any CAD operations or Boolean operations. More complicated shapes such as the BlanketFP() (see Figure **??**) finds points on the front surfaces using a variable offset from the plasma. A variable thickness between the front and rear surface is then used to find the rear surface points. The front and rear surface points area connected with a series of splines with straight connections between the two surfaces. The variable offsets and thicknesses can be provided as a function of poloidal angle and the component is therefore able to construct more complex 3D objects as shown in Figure 2. Some components (e.g. InnerFirstwallFCCS()) are entirely based on other components and finding coordinates to connect up is not necessary as a surface offset and Boolean cut is sufficient to find the 3D volume.

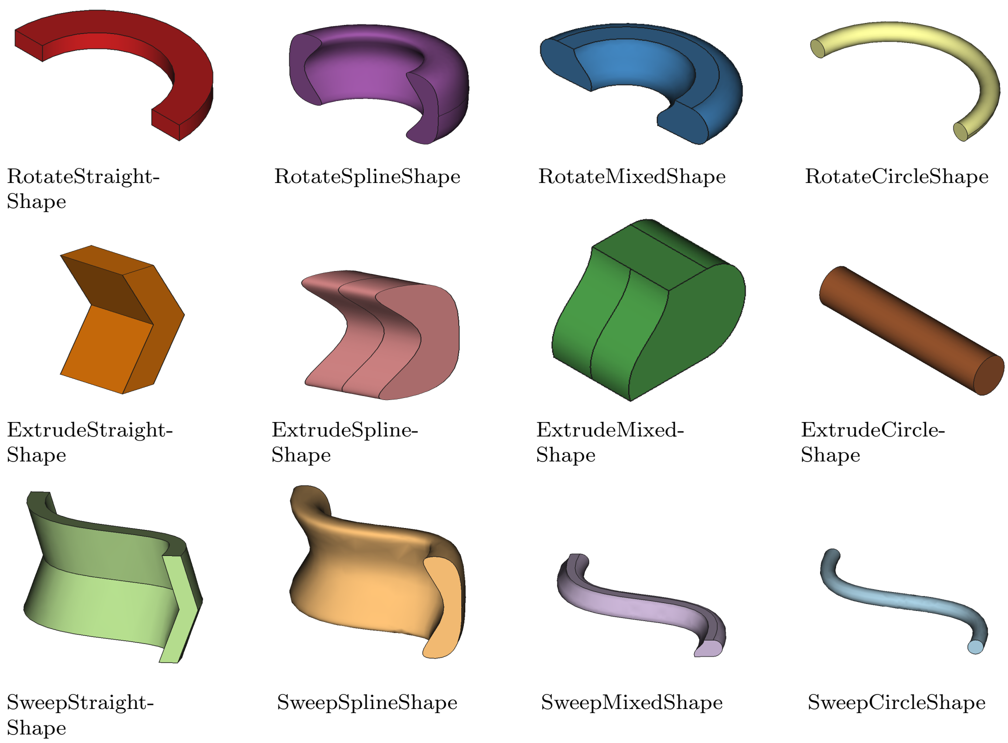


Figure 1: Primitive Shape Classes

There are currently over 34 parametric components available and many additional shapes are planned. When these components are combined then the variety of 3D volumes available is sufficient to start constructing simple fusion reactors as shown in Figure 3.

Parametric reactors combine parametric components and shapes with linkage that describes how they fit together. Parametric reactors allow users to create a 3D reactor model as shown in Figures 6 and 7, these models have been created entirely from parameters. In the case of the SegmentedBallReactor() the model has no inboard breeder zone and has divertors in the upper and lower positions. There are also single-null varieties of the BallReactor(). The main user inputs required are the radial thickness of components. The reactor design has the order of components encoded and therefore from this user information it is possible to know where each component starts and ends in the radial direction. The vertical build for the SegmentedBallReactor() is largely based on the radial build which greatly minimises the number of user inputs required for a 3D model. The user inputs for the plasma elongation and triangularity, combined with the radial build parameters for the plasma, allow the coordinates of the top of the plasma to be calculated. The vertical offset from the firstwall to the plasma defaults to the same value as the outboard plasma gap radial thickness but can be specified independently using the plasma gap vertical thickness parameter. The blanket thickness is constant all around the reactor both in radial and vertical directions. The parametric shape from the blanket accepts a variable thickness as a function of angle (see Figure ??) however this particular reactor design has been programmed to have constant thickness blankets throughout. This means the users will not be asked for a vertical blanket thickness but have less control over the reactor. The blanket is also segmented by another parametric

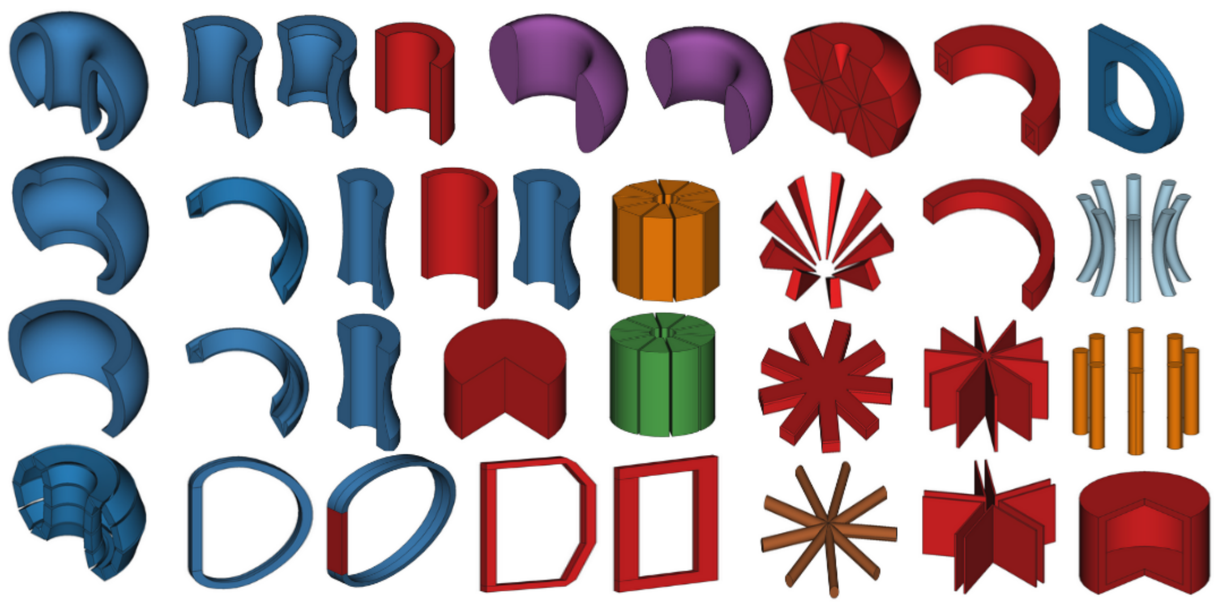


Figure 2: The current selection of parametric components available. Note that because these shapes are all customisable with parameters they can appear differently to their default view pictured in the diagram.

shape (BlanketStarCutter()) to create banana segments. CadQuery's inbuilt filter methods are then used to select the front edges of the firstwall and breeder zone so that they can be filleted. A Boolean cut between the firstwall block and the breeder zone results in a wrap around design. Positioning of poloidal field coils is a user controllable argument, however if (R,Z) coordinates are not specified then they are equispaced vertically behind the blanket. Four types of toroidal field coils exist as parametric components (rectangle, coat hanger, Princeton-D and triple arc) however, simple rectangular toroidal field coils are used for the current BallReactor() design. The SegmentedBallReactor() inherits from the BallReactor() so it also uses rectangular magnets by default. When inheriting from a base design it is possible to overwrite any of the components. Due to this system the number of variations on the base design can rapidly increase. The BallReactor() design has inbuilt assumptions regarding the connections and shapes of components, this has disadvantages in terms of the flexibility but also the advantage of having reduced inputs for the user to specify.

The SubmersionReactor() requires slightly more inputs from the users and offers more flexibility when creating the models. Additional inputs are required for the radial thickness of the supports and the radial thickness of the inner blanket. The computational time to generate the 3D volumes and export CAD files in STP format once the input dimensions have been specified varies from 30 seconds for a simple BallReactor() to 90 seconds for a SegmentedBallReactor() on a standard desktop computer. In this case the time difference is due to segmenting the blanket and filleting the edges of the blanket. Currently the entire construction process is a serial operation so there is scope to speed up the construction by parallelising parts of the construction process.

The CenterColumnStudy() reactor is designed for a specific use case. When performing parametric studies involving the impact of geometry on the center column it is possible to simplify the design to only include components that impact the simulation result. Outboard

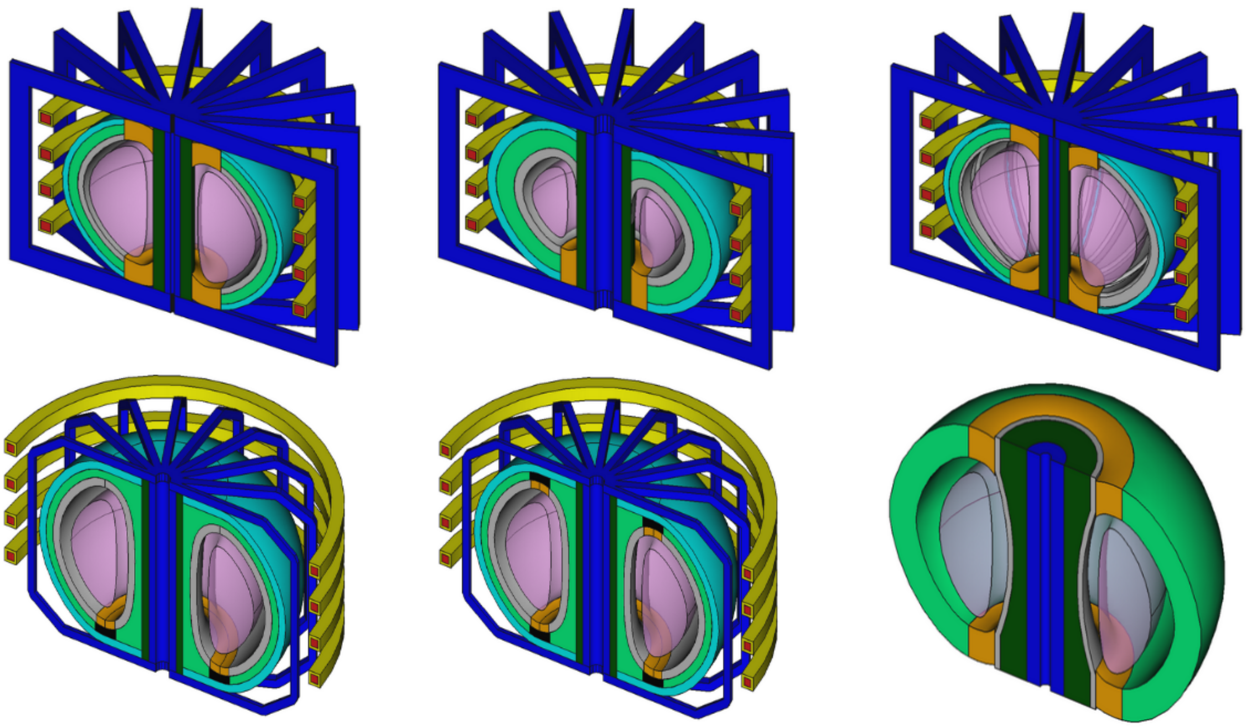


Figure 3: The current selection of reactors available. Note that because these reactors are all customisable with parameters they can appear differently to their default view pictured in the diagram. From left to right and up to down the reactor class names are BallReactor(), SingleNullBallReactor(), SegmentedBallReactor(), SingleNullSubmersionReactor(), SubmersionReactor() and CenterColumnStudyReactor()

TF and PF coils have little impact on the simulation results. This reduces the time needed for model creation and reduces model initialisation in analysis use cases.

While the existing parametric reactors are not a full representation of magnetic fusion reactors the framework established can be used to create more detailed components with more complex relationships between components.

All the various reactor classes allow operations such as exporting the volume(s) to CAD files (STP and STL format) and 2D images (SVG) of the geometry as used in the documentation [7]. Other properties of the geometry can easily be obtained such as the volume of each shape or component in the reactor. This can be useful for cost estimates in systems codes or mass calculations in remote maintenance strategies. The utility of CAD models goes beyond visualisation and basic properties in assessing a design's suitability and can be used as part of an automated parameter study. The Paramak knows the extent of the x, y, z dimensions for the geometry and therefore can automatically create thin shell bounding boxes (referred to as Graveyard volumes) for use in CAD based neutronics with DAGMC [10]. While this paper aims to focus on the geometry creation within the Paramak there are future papers planned where utilisation within neutronics and engineering workflows will be demonstrated.

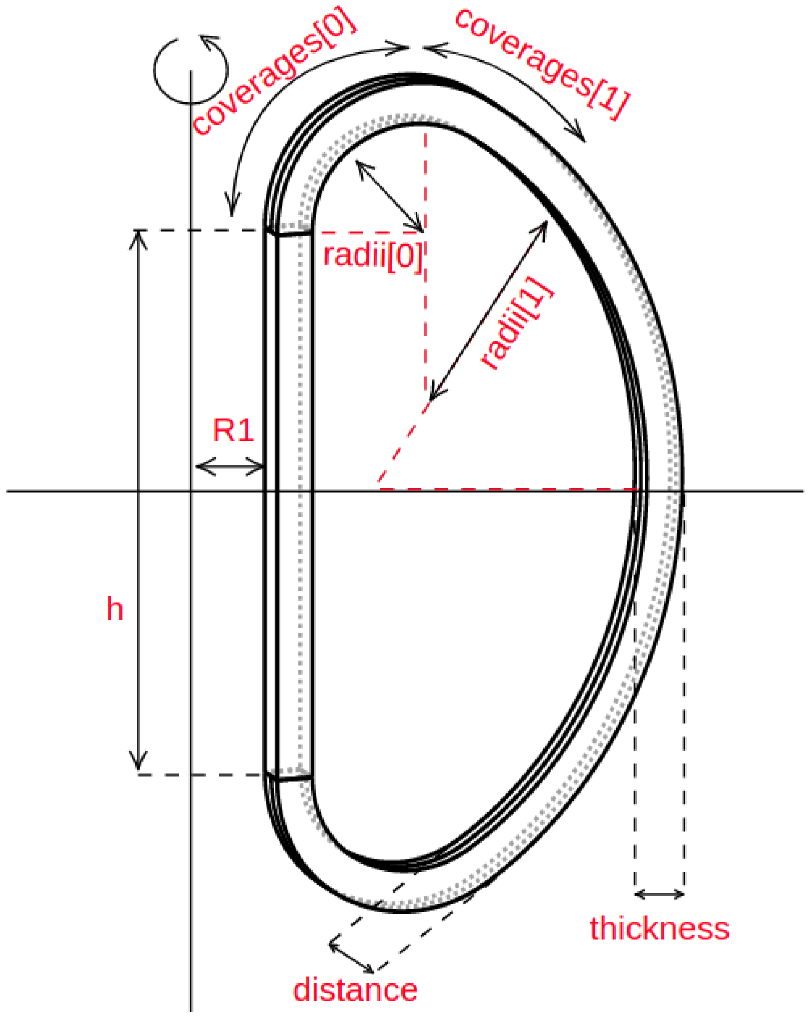
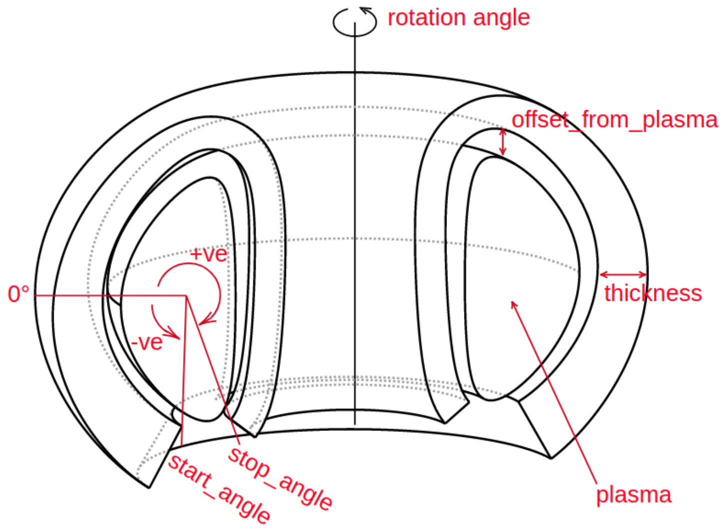


Figure 4: Example parametric component TripleArcTF() with parameters labelled.

4. Conclusion

The Paramak code has been introduced and the motivation of facilitating a data science approach to geometry construction has been discussed. Several benefits of the open-source approach have been realised during the project. The number of parametric components has grown to the level where simplified reactor models can be constructed. Reactor models can be encoded to encapsulate design decisions which allows the required user inputs to be limited. This is demonstrated by the three example models presented in the paper and reinforced by additional parametric reactor models contained in the documentation [7]. There are currently six different parametric reactors for users to create. Due to the structure of the code it is straightforward to inherit existing reactors and modify specific parts of their design to extend the reactor family to accommodate additional features or parameters of parametric reactors.

The current parametric models provided in the Paramak are relatively simple but it is

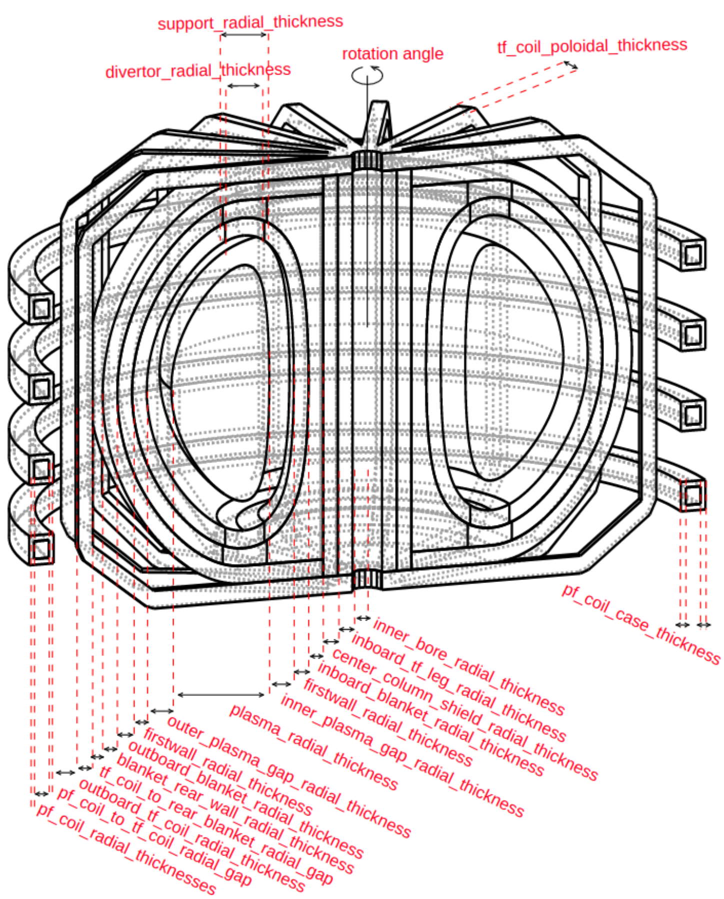



```python
import paramak

plasma = paramak.Plasma(
    elongation=2,
    major_radius=450,
    minor_radius=150,
    triangularity=0.55,
    rotation_angle=180,
)

blanket = paramak.BlanketFP(
    plasma=plasma,
    stop_angle=250,
    start_angle=-70,
    offset_from_plasma=[30,60,30],
    rotation_angle=180,
    thickness=[150,70,70]
)
```

Figure 5: Example parametric component BlanketFP() being build using with a parametric Plasma as one of the inputs. Additionally the blanket has a variable thickness and variable offset from the plasma.

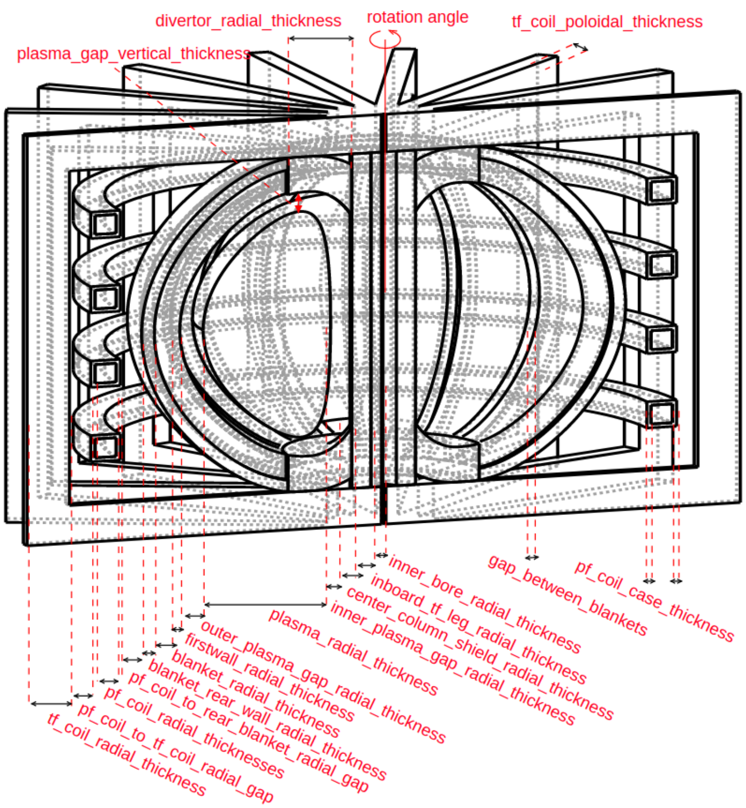



```python
import paramak

my_reactor = paramak.SubmersionTokamak(
      inner_bore_radial_thickness=30,
      inboard_tf_leg_radial_thickness=30,
      center_column_shield_radial_thickness=30,
      divertor_radial_thickness=80,
      inner_plasma_gap_radial_thickness=50,
      plasma_radial_thickness=200,
      outer_plasma_gap_radial_thickness=50,
      firstwall_radial_thickness=30,
      blanket_rear_wall_radial_thickness=30,
      number_of_tf_coils=16,
      rotation_angle=180,
      support_radial_thickness=50,
      inboard_blanket_radial_thickness=30,
      outboard_blanket_radial_thickness=30,
      plasma_high_point=(200, 150),
      pf_coil_radial_thicknesses=[30, 30, 30, 30],
      pf_coil_vertical_thicknesses=[30, 30, 30, 30],
      pf_coil_to_tf_coil_radial_gap=50,
      outboard_tf_coil_radial_thickness=30,
      outboard_tf_coil_poloidal_thickness=30,
      tf_coil_to_rear_blanket_radial_gap=20,
)

my_reactor.export_stp()
my_reactor.export_svg()
```

Figure 6: Example Python script showing the input parameters used for the creation of a SubmersionReactor() model. The example also exports the SVG image used in this Figure and CAD files (STP) used when making Figure 3




```python
import paramak

my_reactor = paramak.SegmentedBlanketBallReactor(
    inner_bore_radial_thickness=5,
    inboard_tf_leg_radial_thickness=25,
    center_column_shield_radial_thickness=45,
    divertor_radial_thickness=150,
    inner_plasma_gap_radial_thickness=50,
    plasma_radial_thickness=300,
    outer_plasma_gap_radial_thickness=50,
    firstwall_radial_thickness=15,
    blanket_radial_thickness=50,
    blanket_rear_wall_radial_thickness=30,
    elongation=2,
    triangularity=0.55,
    number_of_tf_coils=16,
    rotation_angle=180,
    pf_coil_radial_thicknesses=[50, 50, 50, 50],
    pf_coil_vertical_thicknesses=[50, 50, 50, 50],
    pf_coil_to_rear_blanket_radial_gap=50,
    pf_coil_to_tf_coil_radial_gap=50,
    outboard_tf_coil_radial_thickness=100,
    outboard_tf_coil_poloidal_thickness=50,
    gap_between_blankets=30,
    number_of_blanket_segments=15,
    blanket_fillet_radius=15,
)

my_reactor.export_stp()
my_reactor.export_svg()
```

Figure 7: Example Python script showing the input parameters used for the creation of a SegmentedBallReactor() model. The example also exports the SVG image used in this Figure and CAD files (STP) used when making Figure 3

also possible to make more complex models when provided with a design.

The Paramak has been used within UKAEA to create models of several spherical tokamak configurations and has also been used to reproduce a SPARC like design based on the diagrams in [11]. The outputs of the Paramak are CAD models which are useful in fusion analysis disciplines such as Finite Element Analysis (FEA), neutronics, visualisation and even cost models which often require CAD files as an input.

The use of these models in automated workflows has yet to be demonstrated in a publication but this would be the next logical stage in the process and the authors plan to publish a range of use cases for the parametric geometry in the future.

References

[1] B. Mukundakrishnan, N. Rajmohan, D. G. Rajnarayan, S. Fugal, A Script-Based CAD System for Aerodynamic Design.

[2] D. C. Adam urbanczyk, Jeremy Wright, CADQuery, A python parametric CAD scripting framework based on OCCT, 2020 (accessed November 17, 2020). https://github.com/CadQuery/cadquery.

[3] Open Cascade - software development company, 2020 (accessed November 17, 2020). https://www.opencascade.com/.

[4] J. Shimwell, J. Billingsley, R Delaporte-Mathurin et al, Paramak source code Github repository, 2020 (accessed November 17, 2020). https://github.com/ukaea/paramak.

[5] PyPi - The Python Package Index, Paramak PyPi distribution, 2020 (accessed November 17, 2020). https://pypi.org/project/paramak.

[6] Dockerhub, containerized distribution via Dockerhub of the Paramak, 2020 (accessed November 17, 2020). https://hub.docker.com/r/ukaea/paramak.

[7] ReadTheDocs, Paramak API documentation and example usage, 2020 (accessed November 17, 2020). https://paramak.readthedocs.io/en/main/.

[8] PullRequest, Code Review as a Service, 2020 (accessed November 17, 2020). https://www.pullrequest.com/.

[9] Circle CI, CI pipeline for the Paramak, 2020 (accessed November 17, 2020). https://app.circleci.com/pipelines/github/ukaea/paramak?branch=main.

[10] P. P. Wilson, T. J. Tautges, J. A. Kraftcheck, B. M. Smith, D. L. Henderson, Acceleration techniques for the direct use of cad-based geometry in fusion neutronics analysis, Fusion Engineering and Design 85 (2010) 1759 – 1765. Proceedings of the Ninth International Symposium on Fusion Nuclear Technology.

[11] A. J. Creely, M. J. Greenwald, S. B. Ballinger, D. Brunner, J. Canik, J. Doody, T. Fülöp, D. T. Garnier, R. Granetz, T. K. Gray, et al., Overview of the sparc tokamak, Journal of Plasma Physics 86 (2020) 865860502.

5. Acknowledgments

The authors would like to acknowledge the financial support of EPSRC. This work has also been part-funded by the RCUK Energy Programme [grant number EP/I501045/1]. The author would also like to thanks Dr Lidija Shimwell Pasuljevic and Helen Gale.