GeometryConverter

Internal Document

Manual and Notes

Rev. JBO	Date	Description	Author	QC	Approver
01	29.08.2025	First Release	HvW		

TABLE OF CONTENT

1 INTRODUCTION

1.1 General

The GeometryConverter is an Excel and Python-based application that can create, save, load, and assemble MP, TP and Tower geometries from a centrally stored SQL database file. It can detect overlaps and extract a skirt automatically. The assembled structure can be exported in JBOOST, WLGEN, Bladed and Genie/Sesam formats.

1.2 Document Purpose

This document is intended to provide background information, assistance, and guidance for users and potential programmers.

1.3 Revision History

Revision history can be added in this section if required.

1.4 Abbreviations & Definitions

Abbreviations	**Definition**
DNV	Det Norske Veritas
FE	Finite Elements
MP	Monopile
RNA	Rotor Nacelle Assembly
SQL	Structured Query Language
TP	Transition Piece

2 REQUIREMENTS AND INSTALLATION

2.1 Requirements

- Python installed and added to the system PATH
- “python_scripts_path” and “python path” set correctly in ‘GlobalConfig’
- Required Python packages installed via ‘install_requirements’
- Excel macros enabled

2.2 Installation

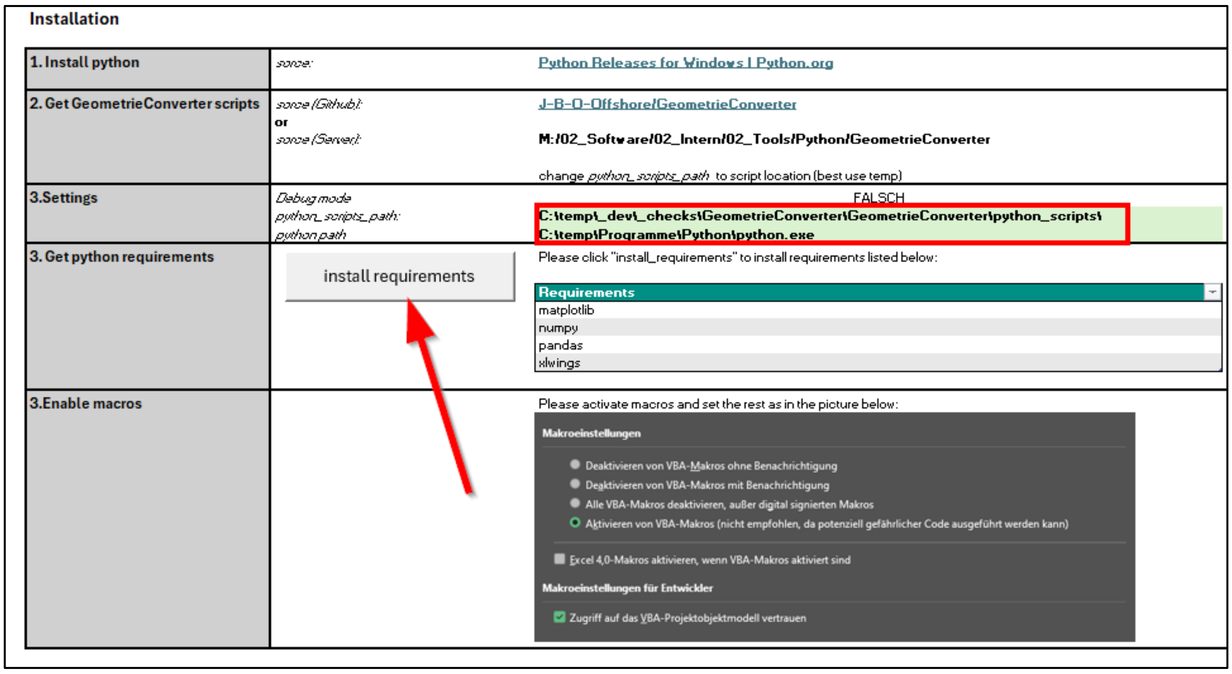
2.2.1 Python.exe and GeometryConverter

1. Install Python Source: Python Releases for Windows → Make sure to add ‘python.exe’ to PATH (can be selected during installation or done afterwards).

2. Get GeometryConverter Scripts Source (GitHub) – python scripts and main excel file with macros:
   - J-B-O-Offshore/GeometrieConverter
     git clone https://github.com/J-B-O-Offshore/GeometrieConverter
     or
   - Server: ‘M:/02_Software/02_Intern/02_Tools/Python/GeometryConverter’

2.2.2 Install Python Requirements for Python Scripts

- Paste your paths in D12 and D13 in ‘GlobalConfig’ sheet from GeometryConverter.xlsm
- Run the batch install_requirements.bat in ‘GlobalConfig’ sheet from GeometryConverter.xlsm



Installation

1. Install python	*sorce:*	Python Releases for Windows I Python.org

2. Get GeometrieConverter scripts	*sorce (Github):*
	or
	sorce (Server):
		J-B-O-Offshore/GeometrieConverter
		
		M:/02_Software/02_Intern/02_Tools/Python/GeometrieConverter
		
		change *python_scripts_path* to script location (best use temp)

3.Settings	*Debug mode* *python_scripts_path:* *python path*	FALSCH **C:\temp_dev_checks\GeometrieConverter\GeometrieConverter\python_scripts** **C:\temp\Programme\Python\python.exe**
3. Get python requirements	install requirements	Please click "install_requirements" to install requirements listed below: **Requirements** matplotlib numpy pandas xlwings
3.Enable macros		Please activate macros and set the rest as in the picture below:

Makroeinstellungen

- ○ Deaktivieren von VBA-Makros ohne Benachrichtigung
- ○ Deaktivieren von VBA-Makros mit Benachrichtigung
- ○ Alle VBA-Makros deaktivieren, außer digital signierten Makros
- ● Aktivieren von VBA-Makros (nicht empfohlen, da potenziell gefährlicher Code ausgeführt werden kann)

☐ Excel 4,0-Makros aktivieren, wenn VBA-Makros aktiviert sind

Makroeinstellungen für Entwickler

☑ Zugriff auf das VBA-Projektobjektmodell vertrauen

Figure 2-1: Enter paths and run “install requirements” in ‘GlobalConfig’ sheet

3 USAGE

Ideally this part should be self-explaining – if not, please provide feedback so that this section gets updated. The excel file consists of 4 sheets:

1. GlobalConfig
2. BuildYourStructure
3. StructureOverview
4. ExportStructure

Figure 3-1: Excel sheets of GeometrieConverter.xlsm

3.1 Excel Sheet “GlobalConfig”

Is for information – just the path in D11 must be set.

3.2 Excel Sheet “BuildYourStructure”

Here the structural data (like diameters, masses,…) of the main parts of the offshore wind turbine (like MP, TP, tower and RNA) can be loaded from databases (file ending *.db). The database file can also be loaded/viewed with the DB Browser for SQLite.

The structural data can be:

- loaded/imported
- adapted/changed
- checked on plausibility
- saved to database
- plotted

3.3 Excel Sheet “StructureOverview”

This is basically a summary (text and graphical) of entered data.

3.4 Excel Sheet “ExportStructure”

Here the defined structural model can be viewed and exported to different formats like:

1. JBOOST – JBO FE tool (C#)
2. WLGEN – JBO wave generator tool (C#)
3. Bladed – DNV software used to design, simulate, and analyze wind turbines, including their structural, mechanical, and aerodynamic behavior
4. Genie/Sesam – DNV’s offshore engineering software suite for modeling, analyzing, and verifying the structural integrity of marine and offshore structures.

4 PYTHON SCRIPTS/FUNCTIONS

4.1 Python File _debug.py – Database Management for Structure Data

This module handles the interaction between Excel (via the *GeometrieConverter.xlsm* workbook) and SQLite databases that store structural geometry and mass data for Monopile (MP), Transition Piece (TP), and Tower elements.

It provides functions to:

- **Read, create, replace, and delete** tables in the database (load_db_table, create_db_table, drop_db_table).
- **Add new structure entries** with geometry, added masses, and metadata (add_db_element).
- **Replace or update existing entries** while checking for duplicates (replace_db_element).
- **Write updated geometry/mass data** back to the database (write_db_element_data).
- **Synchronize Excel dropdowns and tables** with database contents (load_META, load_DATA).
- **Save or update** the selected structure from Excel to the database, with validation and change detection (save_data).
- **Delete** selected structures from the database and clear them from Excel (delete_data).

For convenience, there are dedicated wrappers for each structure type (save_MP_data, save_TP_data, save_TOWER_data, etc.), ensuring the correct database and Excel ranges are used.

In short, _debug.py is the central link that ensures Excel and the SQLite databases remain synchronized, allowing engineers to manage structural data directly from an Excel interface without manual SQL work.

4.2 Python File db_handling.py – Excel-integrated database interface

This module connects the Excel workbook (*GeometrieConverter.xlsm*) with SQLite databases for Monopile (MP), Transition Piece (TP), Tower, and RNA data. It wraps database operations (create, replace, append, delete) with Excel-friendly error messages and automatically syncs dropdowns, tables, and plots.

Key features include:

- Loading, saving, and deleting geometry, masses, and metadata directly from Excel.
- Preventing duplicate entries and validating numeric content.
- Component-specific helpers for MP, TP, and Tower that trigger plotting after loading.
- Importing geometry/masses from external workbooks.
- Managing RNA tables and making them visible in Excel for review.

In short, db_handling.py keeps Excel and the databases synchronized while simplifying structure management without requiring manual SQL.

4.3 Python File excel.py

This module provides high-level helper functions for interacting with open Excel workbooks via **xlwings**, focusing on reading/writing data, controlling form elements, and embedding visual outputs. It supports:

- **Logging setup** for consistent debug and info output.
- **Dropdown and table control** – populate Form Control dropdowns, replace Excel Table contents, clear table bodies, and trigger VBA macros.
- **Data exchange** – write Pandas DataFrames or single values to specific cells, ranges, or named ranges; read tables, ranges, and individual cells back into Pandas; import with optional type casting and NaN handling.
- **User feedback** – display interactive message boxes in Excel using dynamically injected VBA code.
- **Data validation & merging** – add unique rows between DataFrames while ignoring selected columns and handling NaN/None equivalence.
- **Visual embedding** – insert Matplotlib figures directly into Excel at named ranges.

In short, excel.py acts as the bridge between Python's data handling and Excel's interface, enabling automated updates, user interaction, and visual reporting without manual Excel editing.

4.4 Python File export.py

This module converts validated structural geometry and mass data from Excel into **Lua input files** and tabular formats for external engineering tools such as **JBOOST**, **WLGen**, and **Bladed**. It handles:

- **Geometry manipulation** – adding nodes at specific heights, interpolating missing values, and calculating segment weight and center of mass for hollow frustums.
- **Deflection calculation** – computing piecewise-linear deflection profiles across MP, TP, and Tower sections from angles or slopes.
- **JBOOST export** – generating structural and project Lua scripts with nodes, elements, point masses, marine growth layers, waterline insertion, and RNA mass/inertia data.
- **WLGen export** – creating Lua files for wave loading generation, including monopile/transition piece geometry, appurtenances, and skirt modeling.
- **Bladed export** – producing nodes/elements tables with section properties, marine growth, and classified point masses for direct use in Bladed.

In short, export.py transforms the internal structural model into ready-to-use input files for simulation and load analysis software, ensuring all geometry, mass, and environmental parameters are correctly represented.

4.5 Python File misc.py

This module contains helper functions for validating, adjusting, and combining **Monopile (MP)**, **Transition Piece (TP)**, and **Tower** structural data—mainly for offshore wind turbine modeling in Excel. It supports:

- **Data validation** – checks for NaN/non-numeric values and detects gaps or overlaps in section heights.
- **Geometry calculations** – center of mass for hollow frustums, segment weights, and interpolation for inserting new elements at specific heights.
- **Structure assembly** – merges MP, TP, and Tower into a continuous model, resolving overlaps via "Skirt" (extra mass) or "Grout" connections, and adjusting heights for perfect fit.
- **Excel integration** – reads/writes structure and mass tables, updates metadata, and prompts users interactively about height references, RNA selection, and assembly issues.
- **Mass integration** – combines distributed and point masses from all components, adjusting elevations.
- **Node extraction** – derives unique node positions from element data, marking section boundaries ("BORDER").
- **Structure shifting** – moves MP or TP segments and associated masses by a given displacement.

In short, misc.py is the engineering logic core that ensures structural datasets are complete, physically consistent, and ready for export or plotting.

4.6 Python File plot.py

This module creates detailed plots of **Monopile (MP)**, **Transition Piece (TP)**, **Tower**, and full assembled structures directly from Excel data, using **Matplotlib**. It supports:

- **Component plotting** – cylindrical segments ("cans") with optional section numbers, wall thickness, slope, and D/t ratio overlays.
- **Mass visualization** – point and distributed added masses, labeled with names and weights, arranged to avoid label overlap.
- **Assembly views** – color-coded MP/TP/Tower plots with optional skirts, waterline, and seabed markers.
- **Excel integration** – reads geometry/mass tables from the workbook, generates plots, and inserts them back into specified sheets and cell anchors.
- **Overview and section plots** – individual MP, TP, and Tower views, plus a combined "Structure Overview" with metadata-based annotations.

In short, plot.py turns the project's geometry and mass datasets into clear, annotated engineering visuals embedded directly in Excel, aiding design review and documentation.

LIST OF FIGURES