

# Marine and Hydrokinetic Toolkit (MHKit) Workshop

*PRESENTED AT*

PAMEC 2020 San Jose, Costa Rica

*PRESENTED BY*

Katherine Klise and Kelley Ruehl



Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.



# Workshop Overview

Motivation and Development

MHKiT Code Hub and Software

MHKiT Modules and Submodules

Installation and Examples

Future Development and Collaboration

# Motivation

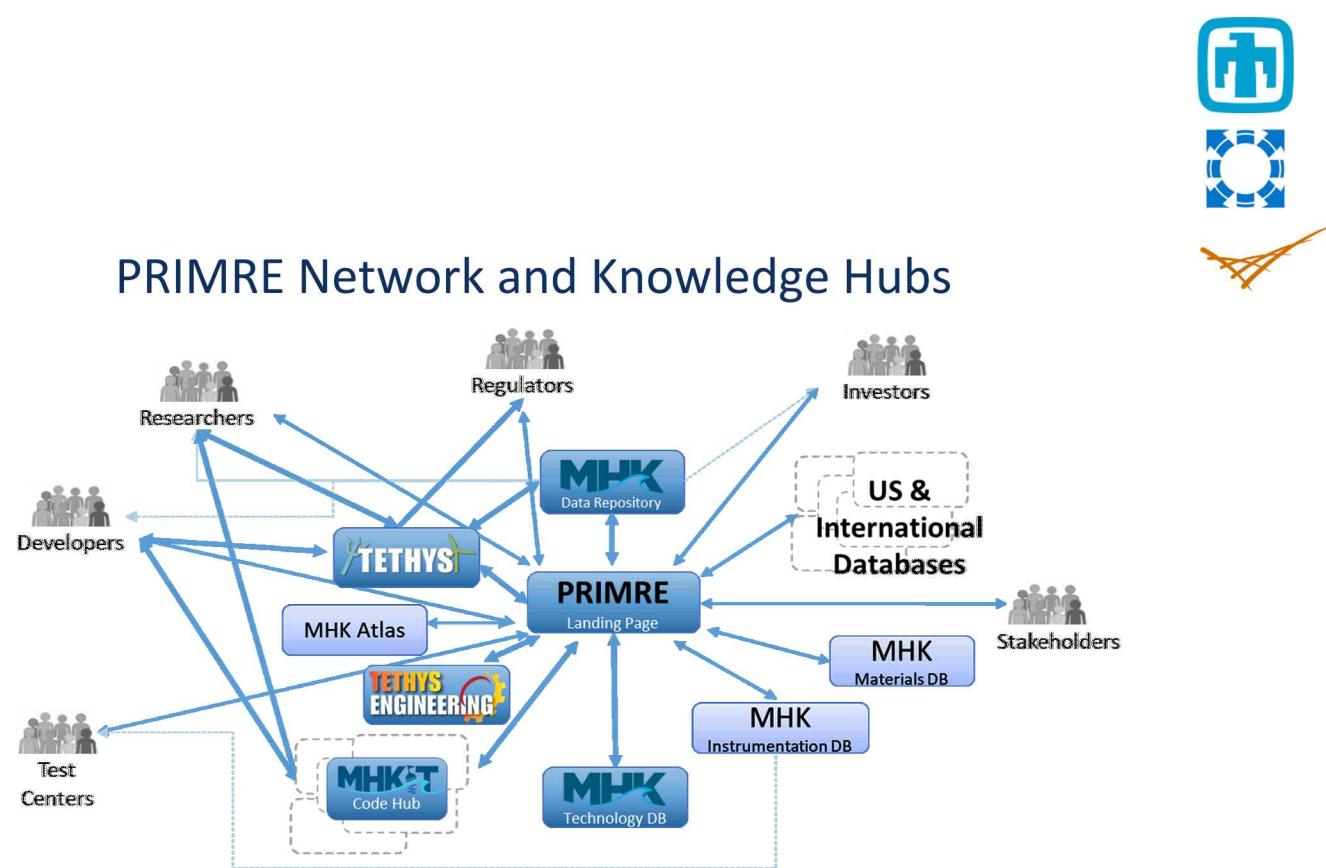
Provide software with the ability to ingest, condition, reduce, quality control, process, visualize and store MRE data.

MHKiT software provides functionality for:

- Data processing (io)
- Data visualization (graphics)
- Data quality control (qc)
- Resource assessment (resource)
- Device performance (performance/device)

The software is developed for MRE data, including measurements from field and laboratory environments, and datasets produced by numerical simulations.

MHKiT is part of the PRIMRE network, as one of the PRIMRE Knowledge Hubs (see network diagram)



## Development Team and Funding



MHKiT is developed as a collaboration between the National Renewable Energy Laboratory (NREL), Pacific Northwest National Laboratory (PNNL), and Sandia National Laboratories (SNL). The core development team is listed below:

- Frederick Driscoll (NREL - PI)
- Budi Gunawan (Sandia - PI)
- Katherine Klise (Sandia)
- Sterling Olson (Sandia)
- Rebecca Pauly (NREL)
- Kelley Ruehl (Sandia)
- Timothy Shippert (PNNL)
- Chitra Sivaraman (PNNL - PI)



MHKiT is funded by the U.S. Department of Energy's Water Power Technologies Office.

<https://mhkit-code-hub.github.io/MHKiT/>

# Licensing



## BSD-3-Clause

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- \* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- \* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- \* Neither the name of Alliance for Sustainable Energy, Battelle Memorial Institute, National Renewable Energy Laboratory, National Technology & Engineering Solutions of Sandia, Pacific Northwest National Laboratory, Sandia National Laboratories, nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## Revised BSD open source license

Copyright 2019, Alliance for Sustainable Energy, LLC under the terms of Contract DE-AC36-08GO28308, Battelle Memorial Institute under the terms of Contract DE-AC05-76RL01830, and National Technology & Engineering Solutions of Sandia, LLC under the terms of Contract DE-NA0003525. The U.S. Government retains certain rights in this software.



Calculations in MHKit adhere to the [International Electrotechnical Committee's Technical Committee's, IEC TC 114](#) technical specifications (TS) and recommendations, as well as follow best practices within the MRE and other fields.

## Wave Module

The Wave module contains a set of functions for wave energy converters (WEC).

The wave module contains the following submodules:

- `io` : Loads wave elevation and resource data.
- `resource` : Computes resource data such as wave height and peak period.
- `performance` : Computes performance metrics for wave energy production. Calculations are based on IEC TS 62600-201:2015 ED1.
- `graphics` : Generates graphics for wave energy converters.

See [MHKit-Python](#) or [MHKit-MATLAB](#) for more details on the wave module.

## Tidal Module

The tidal module contains a set of functions for tidal energy converters (TEC).

The tidal module contains the following submodules:

- `io` : Loads tidal velocity and current data from National Oceanic and Atmospheric Administration (NOAA) current data.
- `resource` : Computes resource data such as directions of flow and directions of current. Calculations are based on IEC TS 62600-201:2015 ED1.
- `device` : Computes device metrics such as equivalent diameter and capture area. Calculations are based on IEC TS 62600-300:2019 ED1.
- `graphics` : Generates graphics for tidal energy converters.

See [MHKit-Python](#) or [MHKit-MATLAB](#) for more details on the tidal module.

## River Module

The river module contains a set of functions to calculate quantities of interest for river energy converters (REC).

The river module contains the following submodules:

- `io` : Loads discharge data from standard formats.
- `resource` : Computes resource assessment metrics, including exceedance probability, inflow velocity, and power (theoretical resource). Calculations are based on IEC TS 62600-301:2019 ED1.
- `device` : Computes device metrics such as equivalent diameter and capture area. Calculations are based on IEC TS 62600-300:2019 ED1.
- `graphics` : Generates graphics, including flow duration curves and velocity duration curves.

See [MHKit-Python](#) or [MHKit-MATLAB](#) for more details on the river module.

# MHKiT Code Hub

MHKiT Code Hub is a collection of MRE software repositories, including:

- **MHKiT-Python**: Software developed in Python
- **MHKiT-MATLAB**: Software developed in MATLAB
- Other relevant open source software:
  - **Pecos**: Data quality control analysis
  - **WEC-Sim**: Wave energy converter simulator
  - **Dolfyn**: Doppler oceanography library
  - ...



<https://github.com/MHKiT-Code-Hub>

 A screenshot of the GitHub repository page for the "MHKiT-Code-Hub" organization. The page shows several pinned repositories: "MHKiT-Python", "MHKiT-MATLAB", "pecos", "WEC-Sim", and "SNL-SWAN". Each repository card includes a brief description, the language it is written in, the license, the number of stars, forks, and commits, and the last update time. A sidebar on the right shows the top languages used in the repositories (Python, Fortran, MATLAB, C++, C) and a list of people associated with the hub.
 

MHKiT Code Hub

Repositories 12 Packages People 1

Pinned repositories

MHKiT-Python MHKiT-Python

Python ★ 3 3

MHKiT-MATLAB MHKiT-MATLAB

MATLAB ★ 2

Find a repository... Type: All Language: All

MHKiT-Python MHKiT-Python

mhkit-python mhkit

Python BSD-3-Clause 3 ★ 3 0 0 Updated 2 hours ago

MHKiT-MATLAB MHKiT-MATLAB

mhkit-matlab mhkit

MATLAB BSD-3-Clause 0 ★ 2 0 0 Updated yesterday

WEC-Sim Forked from WEC-Sim/WEC-Sim

Wave Energy Converter Simulator (WEC-Sim)

MATLAB Apache-2.0 60 ★ 0 0 0 Updated 2 days ago

pecos Forked from sandialabs/pecos

Python package for performance monitoring of time series data

Python 33 ★ 0 0 0 Updated 13 days ago

SNL-SWAN Forked from SNL-WaterPower/SNL-SWAN

SNL-SWAN

Fortran 5 ★ 0 0 0 Updated on Dec 20, 2019

Top languages

Python Fortran MATLAB

C++ C

People 1





MHKit-Python and MHKit-MATLAB are organized into the following modules:

- **QC Module**: Perform quality control analysis
- **Wave Module**: Calculate quantities of interest for wave energy converters (WEC)
- **River Module**: Calculate quantities of interest for river energy converters (REC)
- **Tidal Module**: Calculate quantities of interest for tidal energy converters (TEC)
- **Utils Module**: Includes helper functions

These modules provide functionality for calculating metrics needed by the MRE community as well as those required for conformity with IEC TS and recommendations. MHKit-Python includes continuous integration software tests that are run using [Travis CI](#).

To ensure consistent results between MHKit-Python and MHKit-MATLAB, all functions are written in Python and housed in the MHKit-Python repository. MHKit-MATLAB then wraps these functions so they can be called from MATLAB.

**MHKit-Python and MHKit-MATLAB provide identical functions in each language.**

# 9 Terminology and Units



## Term Definition

$A_P$	Projected capture area [ $m^2$ ]
BS	Bretschneider spectrum
$D_E$	Equivalent diameter [m]
$E$	Energy [J]
$\eta$	Incident wave [m]
$f$	Frequency [Hz]
$F$	Exceedance probability [%]
Fr	Froude Number
$g$	Gravity [ $m/s/s$ ]
$h$	Water depth from bottom to water surface (e.g. SWL) [m]
$H$	Wave height [m]
$H_s$	Significant wave height, mean wave height of the tallest third of waves [m]
$H_{m0}$	Spectrally derived significant wave height [m]
$J$	Wave energy flux [ $W/m$ ]
JS	JONSWAP spectrum
$L$	Capture length [m]

## Term Definition

$k$	Wave number, $k = \frac{2\pi}{\lambda}$ [rad/m]
$m$	Mass [kg]
$m_k$	Spectral moment of $k$ , for $k = 0,1,2,\dots$
$\omega$	Wave frequency, $\omega = \frac{2\pi}{T}$ [rad/s]
$P$	Power [W]
PM	Pierson-Moskowitz spectrum
$Q$	Discharge [ $m^3/s$ ]
$\rho$	Density [ $kg/m^3$ ]
$S$	Spectral density [ $m^2/Hz$ ]
SWL	Still water line
$T_e$	Energy period [s]
$T_m$	Mean wave period [s]
$T_p$	Peak period [s]
$T_z$	Zero-crossing period [s]
$v$	Velocity [ $m/s$ ]
$V$	Velocity calculated for river and tidal modules [ $m/s$ ]

The methods in MHKit use the MKS (meters-kilograms-seconds) system, and assume data is stored in SI units.



The following MHKit demonstration focuses on MHKit-Python

MHKit-Python requires [Python \(3.6 or 3.7\)](#) and has the following Python packages dependencies:

- [Pandas](#): used for data storage and analysis
- [NumPy](#): used for data storage and analysis
- [SciPy](#): used for numerical methods, statistics, and signal processing
- [Matplotlib](#): used to produce figures
- [Requests](#): used to get data from websites
- [Pecos v0.1.8](#): used for quality control analysis

It is recommended to use the [Anaconda Python Distribution](#), since it includes all of the MHKit-Python package dependencies except Pecos.

Users are encouraged to install MHKit-Python using PIP: **pip install mhkit**

<https://mhkit-code-hub.github.io/MHKit/installation.html>

# Getting Started

To get started, import MHKIT-Python

```
import mhkit
```

Access the qc, wave, river, tidal, and utils modules using the following syntax

```
mhkit.qc
mhkit.wave
mhkit.river
mhkit.tidal
mhkit.utils
```

Individual functions are accessed from each module, for example:

```
mhkit.qc.check_timestamp
```

API documentation includes a list of functionality for each module along with details on function input and output

Docs > MHKIT-Python > QC Module [View page source](#)

## QC Module

The QC module includes quality control functions from Pecos, see <https://pecos.readthedocs.io> for more details.

<code>check_timestamp</code>	Check time series for missing, non-monotonic and duplicate timestamps
<code>check_missing</code>	Check for missing data
<code>check_corrupt</code>	Check for corrupt data
<code>check_range</code>	Check for data that is outside expected range
<code>check_delta</code>	Check for staganant data and/or abrupt changes in the data using the difference between max and min values within a rolling window
<code>check_outlier</code>	Check for outliers using normalized data within a rolling window

`mhkit.qc.check_timestamp(data, frequency, expected_start_time=None, expected_end_time=None, min_failures=1, exact_times=True)` [\[source\]](#)

Check time series for missing, non-monotonic and duplicate timestamps

**Parameters**

- `data (pandas DataFrame)` – Data used in the quality control test, indexed by datetime
- `frequency (int)` – Expected time series frequency, in seconds
- `expected_start_time (Timestamp (optional))` – Expected start time. If not specified, the minimum timestamp is used
- `expected_end_time (Timestamp (optional))` – Expected end time. If not specified, the maximum timestamp is used
- `min_failures (int (optional))` – Minimum number of consecutive failures required for reporting, default = 1
- `exact_times (bool (optional))` – Controls how missing times are checked. If True, times are expected to occur at regular intervals (specified in frequency) and the DataFrame is reindexed to match the expected frequency. If False, times only need to occur once or more within each interval (specified in frequency) and the DataFrame is not reindexed.

**Returns**

`dictionary` – Results include cleaned data, mask, and test results summary





The wave, river, and tidal modules are divided into submodules:

## Wave Module

The Wave module contains a set of energy converters (WEC).

The wave module contains the following submodules:

- `io` : Loads wave elevation and direction data.
- `resource` : Computes resource assessment metrics, including wave height and peak period.
- `performance` : Computes performance metrics for wave energy production. Calculations are based on [IEC TS 62600-201:2015 ED1](#).
- `graphics` : Generates graphics, including flow duration curves and velocity duration curves.

See [MHKit-Python](#) or [MHKit-MATLAB](#) for more details on the wave module.

<https://mhkit-code-hub.github.io/MHKit/wave.html>

## Tidal Module

The tidal module contains a set of energy converters (TEC).

The tidal module contains the following submodules:

- `io` : Loads tidal velocity and direction data.
- `resource` : Computes resource assessment metrics, including directions of flow and directions of current. Calculations are based on [IEC TS 62600-201:2015 ED1](#).
- `device` : Computes device metrics such as equivalent diameter and capture area. Calculations are based on [IEC TS 62600-201:2015 ED1](#).
- `graphics` : Generates graphics, including flow duration curves and velocity duration curves.

See [MHKit-Python](#) or [MHKit-MATLAB](#) for more details on the tidal module.

<https://mhkit-code-hub.github.io/MHKit/tidal.html>

## River Module

The river module contains a set of functions to calculate quantities of interest for river energy converters (REC).

The river module contains the following submodules:

- `io` : Loads discharge data from standard formats.
- `resource` : Computes resource assessment metrics, including exceedance probability, inflow velocity, and power (theoretical resource). Calculations are based on [IEC TS 62600-301:2019 ED1](#).
- `device` : Computes device metrics such as equivalent diameter and capture area. Calculations are based on [IEC TS 62600-300:2019 ED1](#).
- `graphics` : Generates graphics, including flow duration curves and velocity duration curves.

See [MHKit-Python](#) or [MHKit-MATLAB](#) for more details on the river module.

<https://mhkit-code-hub.github.io/MHKit/river.html>



MHKit-Python uses Pandas DataFrames to store labelled data structures.

- Each column is labelled with a descriptive name
- Data is indexed by time in seconds or by datetime
- The [Utils Module](#) can be used to convert numeric indexes to datetime indexes

Pandas includes many options including the ability to:

- Read data from various file formats (csv, excel, json, database, etc.)
- Write data and results to various file formats
- Plot timeseries data
- Slice and query data
- Upscale and downscale the time resolution of data
- Fill and interpolate gaps in data
- Compute moving window statistics

Time	probe1	probe2	probe3
10.000	24.48	28.27	1.3
10.002	34.48	40.27	-8.7
10.004	30.48	38.27	-13.7
10.006	12.48	24.27	-32.7
10.008	13.48	22.27	-21.7

Time	probe1	probe2	probe3
2019-05-20 00:00:10.000	24.48	28.27	1.3
2019-05-20 00:00:10.002	34.48	40.27	-8.7
2019-05-20 00:00:10.004	30.48	38.27	-13.7
2019-05-20 00:00:10.006	12.48	24.27	-32.7
2019-05-20 00:00:10.008	13.48	22.27	-21.7

MHKit-Python users that are new to Pandas are encouraged to review the [Pandas getting started guide](#).

# Jupyter Notebook Examples

MHKiT-Python includes four Jupyter Notebook examples to demonstrate functionality, these include:

- **QC example:** run a simple quality control analysis on wave elevation data
- **Wave example:** generate a capture length matrix, calculate MAEP, and plot scatter diagrams
- **River example:** calculate annual energy produced for one turbine in the Tanana river near Nenana, Alaska
- **Tidal example:** calculate velocity duration curve using 1 year of data from the NOAA-Currents sites

## Load Data

The wave elevation data used in this example includes several issues, including timestamps that are out of order, corrupt data with values of -999, data outside expected range, and stagnant data.

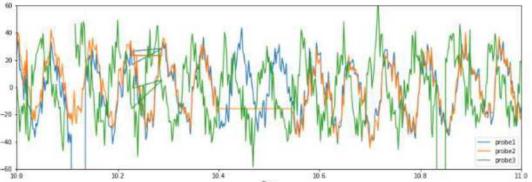
The data is loaded into a pandas DataFrame using the pandas method `read_csv`. The first 5 rows of data are shown below, along with a plot.

```
In [2]: # Load data from the csv file into a DataFrame
data = pd.read_csv('data/wave_elevation_data.csv', index_col='Time')

# Plot the data
data.plot(figsize=(15,5), ylim=(-60,60))

# Print the first 5 rows of data
print(data.head())
```

```
probe1 probe2 probe3
Time
10.000 24.48 28.27 1.3
10.000 34.48 48.27 -8.7
10.004 24.48 28.27 -13.7
10.006 12.48 24.27 -32.7
10.008 13.48 22.27 -21.7
```



The data is indexed by time in seconds. To use the quality control functions, the data must be indexed by datetime. The index can be converted to datetime using the following utility function.

```
In [3]: # Convert the index to datetime
data.index = utils.index_to_datetime(data.index, origin='2010-05-20')

# Print the first 5 rows of data
print(data.head())
```

```
probe1 probe2 probe3
```

```
Time
2010-05-20 00:00:18.000 24.48 28.27 1.3
2010-05-20 00:00:18.002 34.48 48.27 -8.7
2010-05-20 00:00:18.004 30.48 38.27 -13.7
2010-05-20 00:00:18.006 12.48 24.27 -32.7
2010-05-20 00:00:18.008 13.48 22.27 -21.7
```

## Quality control tests

The following quality control tests are used to identify timestamp issues, corrupt data, data outside expected range, and stagnant data.

- Cleaned data, which is a DataFrame that has `NaN` in place of data that did not pass the quality control test.
- Boolean mask, which is a DataFrame with True/False that indicates if each data point passed the quality control test.
- Summary of the quality control test results, the summary includes the variable name (which is blank for timestamp issues), the start and end time of the test failure, and an error flag for each test failure.

## Principal Flow Directions

As an initial check on the data, a velocity plot can be created to identify data gaps. To consider the velocity in one of the principal flow directions we apply the `principal_flow_directions` function. This function returns 2 directions (in degrees) corresponding to the flood and ebb directions of the tidal site. Principal flow directions are calculated based on the highest frequency directions. These directions are often close to 180 degrees apart, but are not required to be.

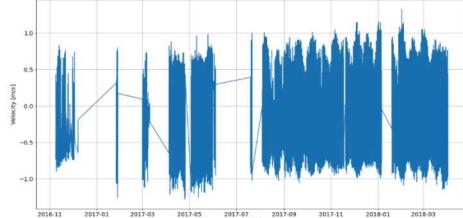
```
In [4]: # Specify histogram bin width for directions to calculate the principal flow directions
width_direction = 1 # in degrees
```

```
# Compute two principal flow directions
direction1, direction2 = tidal.resource.principal_flow_directions(data_d, width_direction)

# Get flood and ebb directions based on site knowledge
flood = direction1 # Flow into
ebb = direction2 # Flow out
```

```
The time series of current data can be plotted using the plot_current_timeseries function, which can include either the flood or ebb directions.
```

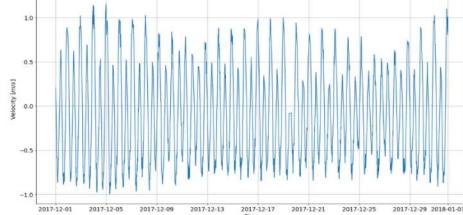
```
In [5]: ax = tidal.graphics.plot_current_timeseries(data_d, data_s, flood)
```



The above plot shows missing data for most of early and mid 2017. The IEC standard recommends a minimum of 1 year of 10 minute averaged data (See IEC 201 for full description). For the purposes of demonstration, this dataset is sufficient. To look at a specific month we can slice the dataset before passing to the plotting function.

```
In [6]: # Slice December data
dec17_data = data.loc['2017-12-01':'2017-12-31']
```

```
# Plot December as current timeseries
ax = tidal.graphics.plot_current_timeseries(dec17_data_d, dec17_data_s, flood)
```

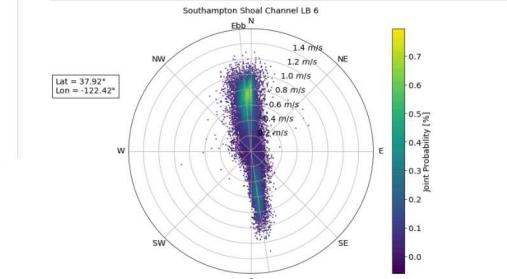


## Joint Probability Distribution

Direction and velocity can be viewed as a joint probability distribution on a polar plot. This plot helps visually show the flood and ebb directions and the frequency of particular directional velocities.

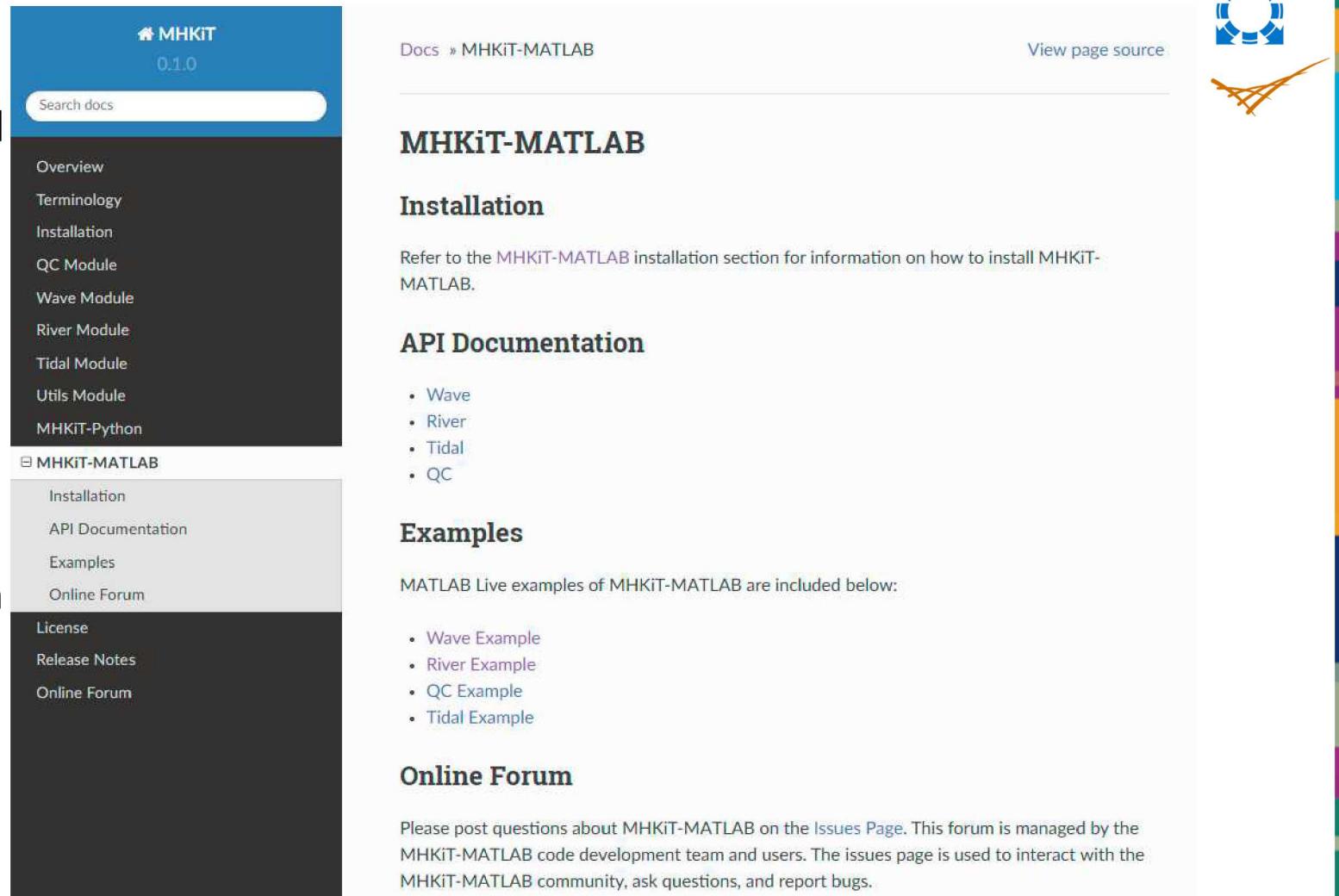
```
In [7]: # Set the joint probability bin widths
width_direction = 1 # in degrees
width_velocity = 0.3 # in m/s

# Plot joint probability distribution
ax = tidal.graphics.plot_joint_probability_distribution(data_d, data_s, width_direction, width_velocity, n_stations=1, etabbb=etabbb, etaflood=flood-ebb)
```



MHKit-MATLAB is intended to be used by researchers and practitioners that prefer MATLAB.

- MHKit-MATLAB runs the MHKit-Python functions by wrapping them in MATLAB.
- The software includes QC, Wave, River, and Tidal modules.
- Documentation includes installation instructions, API documentation, and MATLAB Live examples
- The online forum is hosted on the MHKit-MATLAB GitHub site.



Docs » MHKit-MATLAB [View page source](#)

## MHKit-MATLAB

### Installation

Refer to the MHKit-MATLAB installation section for information on how to install MHKit-MATLAB.

### API Documentation

- Wave
- River
- Tidal
- QC

### Examples

MATLAB Live examples of MHKit-MATLAB are included below:

- Wave Example
- River Example
- QC Example
- Tidal Example

### Online Forum

Please post questions about MHKit-MATLAB on the [Issues Page](#). This forum is managed by the MHKit-MATLAB code development team and users. The issues page is used to interact with the MHKit-MATLAB community, ask questions, and report bugs.

<https://mhkit-code-hub.github.io/MHKit/matlab.html>



## MHKit v0.1.0

The first official release of MHKit, developed in Python and MATLAB, includes the following modules:

- **QC Module:** Perform quality control analysis
- **Wave Module:** Calculate quantities of interest for wave energy converters (WEC)
- **River Module:** Calculate quantities of interest for river energy converters (REC)
- **Tidal Module:** Calculate quantities of interest for tidal energy converters (TEC)
- **Utils Module:** Includes helper functions

The v0.1.0 release includes methods for resource assessment, device performance, graphics, io and quality control.

### [MHKit-Python v0.1.0](#)

### [MHKit-MATLAB v0.1.0](#)

## Future Releases

The next release of MHKit, planned for Summer 2020, will include the following modules:

- **Mechanical Loads Module:** Calculates quantities of interest for mechanical loads.
- **Power Quality Module:** Calculates quantities of interest for power quality.



## Issues

Questions, feature requests, and bug reports for [MHKit-Python](#) and [MHKit-MATLAB](#) should be submitted to the GitHub Issues Page. The GitHub online forums are managed by the MHKit development team and users.

[Submit MHKit-Python Issue](#)

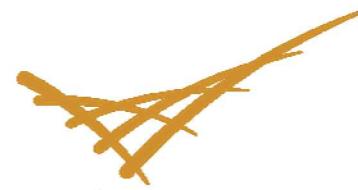
[Submit MHKit-MATLAB Issue](#)

## Collaboration

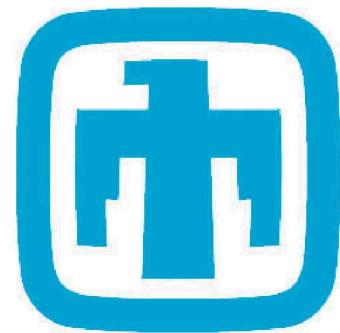
[MHKit-Python](#) and [MHKit-MATLAB](#) welcomes feedback and code contributions. Software developers interested in contributing to the MHKit open-source software are encouraged to use GitHub to create a [Fork](#) of the repository into their GitHub user account. To include your additions to the MHKit code, please submit a [pull request](#) in the master branch of the . Once reviewed by the MHKit development team, pull requests will be merged into MHKit master branch, and included in future releases of MHKit. Software developers, within the MHKit development team and external collaborators, are expected to follow standard practices to document and test new code.

[Submit MHKit-Python Pull Request](#)

[Submit MHKit-MATLAB Pull Request](#)



**Pacific Northwest**  
NATIONAL LABORATORY



**Sandia**  
National  
Laboratories

Thank you!

---

[Katherine.Klise@sandia.gov](mailto:Katherine.Klise@sandia.gov)

[Kelley.Ruehl@sandia.gov](mailto:Kelley.Ruehl@sandia.gov)