windtunnel Documentation

Release 0.0.1

Benyamin Schliffke & Jessica Wiedemeier

Sep 17, 2018

CONTENTS

A collection of tools for basic boundary layer and concentration measurements analysis with Python 3.

CHAPTER ONE

LICENSE

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The GNU General Public License is a free, copyleft license for
software and other kinds of works.

  The licenses for most software and other practical works are designed
to take away your freedom to share and change the works.  By contrast,
the GNU General Public License is intended to guarantee your freedom to
share and change all versions of a program--to make sure it remains free
software for all its users.  We, the Free Software Foundation, use the
GNU General Public License for most of our software; it applies also to
any other work released this way by its authors.  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
them if you wish), that you receive source code or can get it if you
want it, that you can change the software or use pieces of it in new
free programs, and that you know you can do these things.

  To protect your rights, we need to prevent others from denying you
these rights or asking you to surrender the rights.  Therefore, you have
certain responsibilities if you distribute copies of the software, or if
you modify it: responsibilities to respect the freedom of others.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must pass on to the recipients the same
freedoms that you received.  You must make sure that they, too, receive
or can get the source code.  And you must show them these terms so they
know their rights.

  Developers that use the GNU GPL protect your rights with two steps:
(1) assert copyright on the software, and (2) offer you this License
giving you legal permission to copy, distribute and/or modify it.

  For the developers' and authors' protection, the GPL clearly explains
```

(continues on next page)

(continued from previous page)

```
that there is no warranty for this free software.  For both users' and
authors' sake, the GPL requires that modified versions be marked as
changed, so that their problems will not be attributed erroneously to
authors of previous versions.

  Some devices are designed to deny users access to install or run
modified versions of the software inside them, although the manufacturer
can do so.  This is fundamentally incompatible with the aim of
protecting users' freedom to change the software.  The systematic
pattern of such abuse occurs in the area of products for individuals to
use, which is precisely where it is most unacceptable.  Therefore, we
have designed this version of the GPL to prohibit the practice for those
products.  If such problems arise substantially in other domains, we
stand ready to extend this provision to those domains in future versions
of the GPL, as needed to protect the freedom of users.

  Finally, every program is threatened constantly by software patents.
States should not allow patents to restrict development and use of
software on general-purpose computers, but in those that do, we wish to
avoid the special danger that patents applied to a free program could
make it effectively proprietary.  To prevent this, the GPL assures that
patents cannot be used to render the program non-free.

  The precise terms and conditions for copying, distribution and
modification follow.

                       TERMS AND CONDITIONS

  0. Definitions.

  "This License" refers to version 3 of the GNU General Public License.

  "Copyright" also means copyright-like laws that apply to other kinds of
works, such as semiconductor masks.

  "The Program" refers to any copyrightable work licensed under this
License.  Each licensee is addressed as "you".  "Licensees" and
"recipients" may be individuals or organizations.

  To "modify" a work means to copy from or adapt all or part of the work
in a fashion requiring copyright permission, other than the making of an
exact copy.  The resulting work is called a "modified version" of the
earlier work or a work "based on" the earlier work.

  A "covered work" means either the unmodified Program or a work based
on the Program.

  To "propagate" a work means to do anything with it that, without
permission, would make you directly or secondarily liable for
infringement under applicable copyright law, except executing it on a
computer or modifying a private copy.  Propagation includes copying,
distribution (with or without modification), making available to the
public, and in some countries other activities as well.

  To "convey" a work means any kind of propagation that enables other
parties to make or receive copies.  Mere interaction with a user through
a computer network, with no transfer of a copy, is not conveying.
```

(continues on next page)

(continued from previous page)

```
  An interactive user interface displays "Appropriate Legal Notices"
to the extent that it includes a convenient and prominently visible
feature that (1) displays an appropriate copyright notice, and (2)
tells the user that there is no warranty for the work (except to the
extent that warranties are provided), that licensees may convey the
work under this License, and how to view a copy of this License.  If
the interface presents a list of user commands or options, such as a
menu, a prominent item in the list meets this criterion.

  1. Source Code.

  The "source code" for a work means the preferred form of the work
for making modifications to it.  "Object code" means any non-source
form of a work.

  A "Standard Interface" means an interface that either is an official
standard defined by a recognized standards body, or, in the case of
interfaces specified for a particular programming language, one that
is widely used among developers working in that language.

  The "System Libraries" of an executable work include anything, other
than the work as a whole, that (a) is included in the normal form of
packaging a Major Component, but which is not part of that Major
Component, and (b) serves only to enable use of the work with that
Major Component, or to implement a Standard Interface for which an
implementation is available to the public in source code form.  A
"Major Component", in this context, means a major essential component
(kernel, window system, and so on) of the specific operating system
(if any) on which the executable work runs, or a compiler used to
produce the work, or an object code interpreter used to run it.

  The "Corresponding Source" for a work in object code form means all
the source code needed to generate, install, and (for an executable
work) run the object code and to modify the work, including scripts to
control those activities.  However, it does not include the work's
System Libraries, or general-purpose tools or generally available free
programs which are used unmodified in performing those activities but
which are not part of the work.  For example, Corresponding Source
includes interface definition files associated with source files for
the work, and the source code for shared libraries and dynamically
linked subprograms that the work is specifically designed to require,
such as by intimate data communication or control flow between those
subprograms and other parts of the work.

  The Corresponding Source need not include anything that users
can regenerate automatically from other parts of the Corresponding
Source.

  The Corresponding Source for a work in source code form is that
same work.

  2. Basic Permissions.

  All rights granted under this License are granted for the term of
copyright on the Program, and are irrevocable provided the stated
conditions are met.  This License explicitly affirms your unlimited
```

(continues on next page)

(continued from previous page)

```
permission to run the unmodified Program.  The output from running a
covered work is covered by this License only if the output, given its
content, constitutes a covered work.  This License acknowledges your
rights of fair use or other equivalent, as provided by copyright law.

  You may make, run and propagate covered works that you do not
convey, without conditions so long as your license otherwise remains
in force.  You may convey covered works to others for the sole purpose
of having them make modifications exclusively for you, or provide you
with facilities for running those works, provided that you comply with
the terms of this License in conveying all material for which you do
not control copyright.  Those thus making or running the covered works
for you must do so exclusively on your behalf, under your direction
and control, on terms that prohibit them from making any copies of
your copyrighted material outside their relationship with you.

  Conveying under any other circumstances is permitted solely under
the conditions stated below.  Sublicensing is not allowed; section 10
makes it unnecessary.

  3. Protecting Users' Legal Rights From Anti-Circumvention Law.

  No covered work shall be deemed part of an effective technological
measure under any applicable law fulfilling obligations under article
11 of the WIPO copyright treaty adopted on 20 December 1996, or
similar laws prohibiting or restricting circumvention of such
measures.

  When you convey a covered work, you waive any legal power to forbid
circumvention of technological measures to the extent such circumvention
is effected by exercising rights under this License with respect to
the covered work, and you disclaim any intention to limit operation or
modification of the work as a means of enforcing, against the work's
users, your or third parties' legal rights to forbid circumvention of
technological measures.

  4. Conveying Verbatim Copies.

  You may convey verbatim copies of the Program's source code as you
receive it, in any medium, provided that you conspicuously and
appropriately publish on each copy an appropriate copyright notice;
keep intact all notices stating that this License and any
non-permissive terms added in accord with section 7 apply to the code;
keep intact all notices of the absence of any warranty; and give all
recipients a copy of this License along with the Program.

  You may charge any price or no price for each copy that you convey,
and you may offer support or warranty protection for a fee.

  5. Conveying Modified Source Versions.

  You may convey a work based on the Program, or the modifications to
produce it from the Program, in the form of source code under the
terms of section 4, provided that you also meet all of these conditions:

    a) The work must carry prominent notices stating that you modified
    it, and giving a relevant date.
```

(continues on next page)

(continued from previous page)

```
    b) The work must carry prominent notices stating that it is
    released under this License and any conditions added under section
    7.  This requirement modifies the requirement in section 4 to
    "keep intact all notices".

    c) You must license the entire work, as a whole, under this
    License to anyone who comes into possession of a copy.  This
    License will therefore apply, along with any applicable section 7
    additional terms, to the whole of the work, and all its parts,
    regardless of how they are packaged.  This License gives no
    permission to license the work in any other way, but it does not
    invalidate such permission if you have separately received it.

    d) If the work has interactive user interfaces, each must display
    Appropriate Legal Notices; however, if the Program has interactive
    interfaces that do not display Appropriate Legal Notices, your
    work need not make them do so.

  A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
"aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users
beyond what the individual works permit.  Inclusion of a covered work
in an aggregate does not cause this License to apply to the other
parts of the aggregate.

  6. Conveying Non-Source Forms.

  You may convey a covered work in object code form under the terms
of sections 4 and 5, provided that you also convey the
machine-readable Corresponding Source under the terms of this License,
in one of these ways:

    a) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by the
    Corresponding Source fixed on a durable physical medium
    customarily used for software interchange.

    b) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by a
    written offer, valid for at least three years and valid for as
    long as you offer spare parts or customer support for that product
    model, to give anyone who possesses the object code either (1) a
    copy of the Corresponding Source for all the software in the
    product that is covered by this License, on a durable physical
    medium customarily used for software interchange, for a price no
    more than your reasonable cost of physically performing this
    conveying of source, or (2) access to copy the
    Corresponding Source from a network server at no charge.

    c) Convey individual copies of the object code with a copy of the
    written offer to provide the Corresponding Source.  This
    alternative is allowed only occasionally and noncommercially, and
    only if you received the object code with such an offer, in accord
```

(continues on next page)

(continued from previous page)

```
    with subsection 6b.

    d) Convey the object code by offering access from a designated
    place (gratis or for a charge), and offer equivalent access to the
    Corresponding Source in the same way through the same place at no
    further charge.  You need not require recipients to copy the
    Corresponding Source along with the object code.  If the place to
    copy the object code is a network server, the Corresponding Source
    may be on a different server (operated by you or a third party)
    that supports equivalent copying facilities, provided you maintain
    clear directions next to the object code saying where to find the
    Corresponding Source.  Regardless of what server hosts the
    Corresponding Source, you remain obligated to ensure that it is
    available for as long as needed to satisfy these requirements.

    e) Convey the object code using peer-to-peer transmission, provided
    you inform other peers where the object code and Corresponding
    Source of the work are being offered to the general public at no
    charge under subsection 6d.

  A separable portion of the object code, whose source code is excluded
from the Corresponding Source as a System Library, need not be
included in conveying the object code work.

  A "User Product" is either (1) a "consumer product", which means any
tangible personal property which is normally used for personal, family,
or household purposes, or (2) anything designed or sold for incorporation
into a dwelling.  In determining whether a product is a consumer product,
doubtful cases shall be resolved in favor of coverage.  For a particular
product received by a particular user, "normally used" refers to a
typical or common use of that class of product, regardless of the status
of the particular user or of the way in which the particular user
actually uses, or expects or is expected to use, the product.  A product
is a consumer product regardless of whether the product has substantial
commercial, industrial or non-consumer uses, unless such uses represent
the only significant mode of use of the product.

  "Installation Information" for a User Product means any methods,
procedures, authorization keys, or other information required to install
and execute modified versions of a covered work in that User Product from
a modified version of its Corresponding Source.  The information must
suffice to ensure that the continued functioning of the modified object
code is in no case prevented or interfered with solely because
modification has been made.

  If you convey an object code work under this section in, or with, or
specifically for use in, a User Product, and the conveying occurs as
part of a transaction in which the right of possession and use of the
User Product is transferred to the recipient in perpetuity or for a
fixed term (regardless of how the transaction is characterized), the
Corresponding Source conveyed under this section must be accompanied
by the Installation Information.  But this requirement does not apply
if neither you nor any third party retains the ability to install
modified object code on the User Product (for example, the work has
been installed in ROM).

  The requirement to provide Installation Information does not include a
```

(continues on next page)

(continued from previous page)

```
requirement to continue to provide support service, warranty, or updates
for a work that has been modified or installed by the recipient, or for
the User Product in which it has been modified or installed.  Access to a
network may be denied when the modification itself materially and
adversely affects the operation of the network or violates the rules and
protocols for communication across the network.

  Corresponding Source conveyed, and Installation Information provided,
in accord with this section must be in a format that is publicly
documented (and with an implementation available to the public in
source code form), and must require no special password or key for
unpacking, reading or copying.

  7. Additional Terms.

  "Additional permissions" are terms that supplement the terms of this
License by making exceptions from one or more of its conditions.
Additional permissions that are applicable to the entire Program shall
be treated as though they were included in this License, to the extent
that they are valid under applicable law.  If additional permissions
apply only to part of the Program, that part may be used separately
under those permissions, but the entire Program remains governed by
this License without regard to the additional permissions.

  When you convey a copy of a covered work, you may at your option
remove any additional permissions from that copy, or from any part of
it.  (Additional permissions may be written to require their own
removal in certain cases when you modify the work.)  You may place
additional permissions on material, added by you to a covered work,
for which you have or can give appropriate copyright permission.

  Notwithstanding any other provision of this License, for material you
add to a covered work, you may (if authorized by the copyright holders of
that material) supplement the terms of this License with terms:

    a) Disclaiming warranty or limiting liability differently from the
    terms of sections 15 and 16 of this License; or

    b) Requiring preservation of specified reasonable legal notices or
    author attributions in that material or in the Appropriate Legal
    Notices displayed by works containing it; or

    c) Prohibiting misrepresentation of the origin of that material, or
    requiring that modified versions of such material be marked in
    reasonable ways as different from the original version; or

    d) Limiting the use for publicity purposes of names of licensors or
    authors of the material; or

    e) Declining to grant rights under trademark law for use of some
    trade names, trademarks, or service marks; or

    f) Requiring indemnification of licensors and authors of that
    material by anyone who conveys the material (or modified versions of
    it) with contractual assumptions of liability to the recipient, for
    any liability that these contractual assumptions directly impose on
    those licensors and authors.
```

(continues on next page)

(continued from previous page)

```
  All other non-permissive additional terms are considered "further
restrictions" within the meaning of section 10.  If the Program as you
received it, or any part of it, contains a notice stating that it is
governed by this License along with a term that is a further
restriction, you may remove that term.  If a license document contains
a further restriction but permits relicensing or conveying under this
License, you may add to a covered work material governed by the terms
of that license document, provided that the further restriction does
not survive such relicensing or conveying.

  If you add terms to a covered work in accord with this section, you
must place, in the relevant source files, a statement of the
additional terms that apply to those files, or a notice indicating
where to find the applicable terms.

  Additional terms, permissive or non-permissive, may be stated in the
form of a separately written license, or stated as exceptions;
the above requirements apply either way.

  8. Termination.

  You may not propagate or modify a covered work except as expressly
provided under this License.  Any attempt otherwise to propagate or
modify it is void, and will automatically terminate your rights under
this License (including any patent licenses granted under the third
paragraph of section 11).

  However, if you cease all violation of this License, then your
license from a particular copyright holder is reinstated (a)
provisionally, unless and until the copyright holder explicitly and
finally terminates your license, and (b) permanently, if the copyright
holder fails to notify you of the violation by some reasonable means
prior to 60 days after the cessation.

  Moreover, your license from a particular copyright holder is
reinstated permanently if the copyright holder notifies you of the
violation by some reasonable means, this is the first time you have
received notice of violation of this License (for any work) from that
copyright holder, and you cure the violation prior to 30 days after
your receipt of the notice.

  Termination of your rights under this section does not terminate the
licenses of parties who have received copies or rights from you under
this License.  If your rights have been terminated and not permanently
reinstated, you do not qualify to receive new licenses for the same
material under section 10.

  9. Acceptance Not Required for Having Copies.

  You are not required to accept this License in order to receive or
run a copy of the Program.  Ancillary propagation of a covered work
occurring solely as a consequence of using peer-to-peer transmission
to receive a copy likewise does not require acceptance.  However,
nothing other than this License grants you permission to propagate or
modify any covered work.  These actions infringe copyright if you do
not accept this License.  Therefore, by modifying or propagating a
```

(continues on next page)

(continued from previous page)

```
covered work, you indicate your acceptance of this License to do so.

  10. Automatic Licensing of Downstream Recipients.

  Each time you convey a covered work, the recipient automatically
receives a license from the original licensors, to run, modify and
propagate that work, subject to this License.  You are not responsible
for enforcing compliance by third parties with this License.

  An "entity transaction" is a transaction transferring control of an
organization, or substantially all assets of one, or subdividing an
organization, or merging organizations.  If propagation of a covered
work results from an entity transaction, each party to that
transaction who receives a copy of the work also receives whatever
licenses to the work the party's predecessor in interest had or could
give under the previous paragraph, plus a right to possession of the
Corresponding Source of the work from the predecessor in interest, if
the predecessor has it or can get it with reasonable efforts.

  You may not impose any further restrictions on the exercise of the
rights granted or affirmed under this License.  For example, you may
not impose a license fee, royalty, or other charge for exercise of
rights granted under this License, and you may not initiate litigation
(including a cross-claim or counterclaim in a lawsuit) alleging that
any patent claim is infringed by making, using, selling, offering for
sale, or importing the Program or any portion of it.

  11. Patents.

  A "contributor" is a copyright holder who authorizes use under this
License of the Program or a work on which the Program is based.  The
work thus licensed is called the contributor's "contributor version".

  A contributor's "essential patent claims" are all patent claims
owned or controlled by the contributor, whether already acquired or
hereafter acquired, that would be infringed by some manner, permitted
by this License, of making, using, or selling its contributor version,
but do not include claims that would be infringed only as a
consequence of further modification of the contributor version.  For
purposes of this definition, "control" includes the right to grant
patent sublicenses in a manner consistent with the requirements of
this License.

  Each contributor grants you a non-exclusive, worldwide, royalty-free
patent license under the contributor's essential patent claims, to
make, use, sell, offer for sale, import and otherwise run, modify and
propagate the contents of its contributor version.

  In the following three paragraphs, a "patent license" is any express
agreement or commitment, however denominated, not to enforce a patent
(such as an express permission to practice a patent or covenant not to
sue for patent infringement).  To "grant" such a patent license to a
party means to make such an agreement or commitment not to enforce a
patent against the party.

  If you convey a covered work, knowingly relying on a patent license,
and the Corresponding Source of the work is not available for anyone
```

(continues on next page)

(continued from previous page)

```
to copy, free of charge and under the terms of this License, through a
publicly available network server or other readily accessible means,
then you must either (1) cause the Corresponding Source to be so
available, or (2) arrange to deprive yourself of the benefit of the
patent license for this particular work, or (3) arrange, in a manner
consistent with the requirements of this License, to extend the patent
license to downstream recipients.  "Knowingly relying" means you have
actual knowledge that, but for the patent license, your conveying the
covered work in a country, or your recipient's use of the covered work
in a country, would infringe one or more identifiable patents in that
country that you have reason to believe are valid.

  If, pursuant to or in connection with a single transaction or
arrangement, you convey, or propagate by procuring conveyance of, a
covered work, and grant a patent license to some of the parties
receiving the covered work authorizing them to use, propagate, modify
or convey a specific copy of the covered work, then the patent license
you grant is automatically extended to all recipients of the covered
work and works based on it.

  A patent license is "discriminatory" if it does not include within
the scope of its coverage, prohibits the exercise of, or is
conditioned on the non-exercise of one or more of the rights that are
specifically granted under this License.  You may not convey a covered
work if you are a party to an arrangement with a third party that is
in the business of distributing software, under which you make payment
to the third party based on the extent of your activity of conveying
the work, and under which the third party grants, to any of the
parties who would receive the covered work from you, a discriminatory
patent license (a) in connection with copies of the covered work
conveyed by you (or copies made from those copies), or (b) primarily
for and in connection with specific products or compilations that
contain the covered work, unless you entered into that arrangement,
or that patent license was granted, prior to 28 March 2007.

  Nothing in this License shall be construed as excluding or limiting
any implied license or other defenses to infringement that may
otherwise be available to you under applicable patent law.

  12. No Surrender of Others' Freedom.

  If conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot convey a
covered work so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you may
not convey it at all.  For example, if you agree to terms that obligate you
to collect a royalty for further conveying from those to whom you convey
the Program, the only way you could satisfy both those terms and this
License would be to refrain entirely from conveying the Program.

  13. Use with the GNU Affero General Public License.

  Notwithstanding any other provision of this License, you have
permission to link or combine any covered work with a work licensed
under version 3 of the GNU Affero General Public License into a single
combined work, and to convey the resulting work.  The terms of this
```

(continues on next page)

(continued from previous page)

```
License will continue to apply to the part which is the covered work,
but the special requirements of the GNU Affero General Public License,
section 13, concerning interaction through a network will apply to the
combination as such.

  14. Revised Versions of this License.

  The Free Software Foundation may publish revised and/or new versions of
the GNU General Public License from time to time.  Such new versions will
be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.

  Each version is given a distinguishing version number.  If the
Program specifies that a certain numbered version of the GNU General
Public License "or any later version" applies to it, you have the
option of following the terms and conditions either of that numbered
version or of any later version published by the Free Software
Foundation.  If the Program does not specify a version number of the
GNU General Public License, you may choose any version ever published
by the Free Software Foundation.

  If the Program specifies that a proxy can decide which future
versions of the GNU General Public License can be used, that proxy's
public statement of acceptance of a version permanently authorizes you
to choose that version for the Program.

  Later license versions may give you additional or different
permissions.  However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow a
later version.

  15. Disclaimer of Warranty.

  THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW.  EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM
IS WITH YOU.  SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  16. Limitation of Liability.

  IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS
THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY
GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE
USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF
DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD
PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS),
EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.

  17. Interpretation of Sections 15 and 16.

  If the disclaimer of warranty and limitation of liability provided
```

(continues on next page)

(continued from previous page)

```
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program>  Copyright (C) <year>  <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, your program's commands
might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see
<http://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program
into proprietary programs.  If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
```

(continues on next page)

(continued from previous page)

```
the library.  If this is what you want to do, use the GNU Lesser General
Public License instead of this License.  But first, please read
<http://www.gnu.org/philosophy/why-not-lgpl.html>.
```

CHAPTER

TWO

CONTRIBUTOR COVENANT CODE OF CONDUCT

2.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

2.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

2.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

2.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

2.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at benny.schliffke@gmail.com. The project team will review and investigate all complaints, and will respond in a way that it deems appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

2.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at http://contributor-covenant.org/version/1/4

CHAPTER

THREE

WINDTUNNEL

Python package for use with output from flow and/or concentration windtunnel measurements.

3.1 Basics

The package has three branches. utils, stats and plots. utils contains utility and support functions for windtunnel timeseries analysis. stats contains functions to calculate turbulence quantities of timeseries' and basic statistical analysis tools. plots has two sub-branches, one for boundary layer analysis (bl) and one containing a few useful plotting tools (tools). The log file is saved to the working directory.

3.2 The Timeseries class

The class Timeseries, seperate from the three branches, holds the raw timeseries with all attributes of the class being defining quantities related to each timeseries (coordinates, wtref, mean wind magnitude, mean wind direction, the measured wind components with their respective timeseries, as well as a transit time weighted mean and variance). The class expects data in the standard BSA software output format. Timeseries inherits from pandas.DataFrame, thus it has all the same funcionality as DataFrame on top of its more specific windtunnel methods. Timeseries includes methods to read data, make the timeseries equisitant, nondimensionalise the timeseries, adapt the scale, mask outliers and calculate wind magnitude and wind direction from the components given. It is also possible to save the manipulated raw timeseries of a Timeseries object.

3.2.1 Example of intended use (Timeseries class)

```
# Input paths for data and wtref with a list of names of the measurement files
path = '/path/to/your/data/'
wtref_path = '/path/to/your/wtref/'
namelist = ['name_of measurement_file']

# Create dictionary for each file in namelist
time_series = {}
time_series.fromkeys(namelist)

# Gather all files into Timeseries objects, manipulate and save raw timeseries
# as txt output and into the dictionary 'time_series'
for name in namelist:
    files = wt.get_files(path,name)
    time_series[name] = {}
    time_series[name].fromkeys(files)
```

(continues on next page)

(continued from previous page)

```
    for i,file in enumerate(files):
        ts = wt.Timeseries.from_file(path+file)
        ts.get_wind_comps(path+file)
        ts.get_wtref(wtref_path,name,index=i)
        ts.nondimensionalise()
        ts.adapt_scale(scale)
        ts.equidistant()
        ts.mask_outliers()
        ts.weighted_component_mean
        ts.weighted_component_variance
        ts.mean_magnitude
        ts.mean_direction
        ts.save2file(file)
        time_series[name][file] = ts
```

3.3 The script 'example_data_analysis.py'

The script 'data_analysis.py' offers a basic boundary analysis based on functions from this package. It offers four different modes of analysis (1 = horizontal profile, 2 = lateral profile, 3 = convergence test, 4 = Reynolds Number Independence). The output type of the images can be specified to any type supported by python. It necessary to specify the paths to the data and wtref as well as the desired output paths for plots and txt files. A geometric scale needs to be defined in order to transfer the results to full-scale coordinates.

3.3.1 Example of 'example_data_analysis.py' input

```
# Input paths for data and wtref with a list of names of the measurement files
path = '/path/to/your/data/'
wtref_path = '/path/to/your/wtref/'
namelist = ['name_of measurement_file']

# Output paths, using the users ID to create a standard path and image output type
plot_path = './plots/'
txt_path = './postprocessed/'
file_type = 'pdf' # (or 'png' etc.)

# Scale and mode desired for the analysis
scale = 500
#1 = vertical profile
#2 = lateral profile
#3 = convergence test
#4 = Reynolds Number Independence
mode = 1
```

3.4 Installing the windtunnel package

The easiest way to work and develop the windtunnel package is to clone the project and install it using pip:

```
$ git clone https://github.com/bschliffke/windtunnel.git
$ cd windtunnel
$ pip install --editable .
```

The `--editable` flag ensures that changes to project files directly affect the package's behaviour in the Python environment.

For Windows users, who are not familiar mit pip on Windows, you can revert to the 'quick and dirty' method. Copy the windtunnel file (and example_data_analysis.py, if required) to your working directory. WARNING! Doing this leaves it up to the user to install all missing dependencies (ie. required packages for proper functionality of windtunnel).

3.5 Useful information

In order to see the docstring and information on the parameters expected by a function, call [functionname]? in the console. Example:

```
In [1]: wt.calc_turb_data?
Signature: wt.calc_turb_data(u_comp, v_comp)
Docstring:
Calculate turbulence intensity and turbulent fluxes from equidistant
times series of u and v components.
@parameter: u_comp: np.array or list
@parameter: v_comp: np.array or list
File:      c:\users\u300517\documents\github\windtunnel\windtunnel\stats.py
Type:      function
```

3.6 Future development

Future development should include a parallel set of function for measurements done in non-coincidence mode. Also a new branch for a quick basic analysis of concentration measurements would be useful. Some open TODOs can be found in windtunnel_playground.py. Any functions developed by single users outside of this package, but are considered useful to the user base of the windtunnel package, may be added. At this point the python PEP 8 – Style Guide for Python Code (https://www.python.org/dev/peps/pep-0008/#code-lay-out) has to be followed to maintain readability and consistency within the package's source code. All maintenance work has to be documented with reasons given for the work done.

3.7 Documentation

Documentation can be found in windtunnel.pdf.

CHAPTER

FOUR

CLASSES AND FUNCTIONS

Python package for basic boundary layer and concentration measurement analysis.

class windtunnel.**PointConcentration**(*time*, *wtref*, *slow_FID*, *fast_FID*, *open_rate*)

PointConcentration is a class that holds data collected during a continuous release point concentration measurement. The class can hold the raw time series, the corresponding wtref and all other quantities necessary to analyse the time series. All the information in a PointConcentration object can be saved to a txt file. @parameter: time, type = np.array @parameter: wtref, type = np.array @parameter: fast_FID, type = np.array @parameter: slow_FID, type = np.array @parameter: open_rate, type = np.array

ambient_conditions(*x*, *y*, *z*, *pressure*, *temperature*, *calibration_curve*, *mass_flow_controller*, *calibration_factor=0*)

Collect ambient conditions during measurement. pressure in [Pa], temperature in [°C].

calc_c_star()

Calculate dimensionless concentration. [-]

calc_full_scale_concentration()

Calculate full scale concentration in [ppmV].

calc_full_scale_flow_rate()

Convert flow rate to full scale flow rate in [m^3/s].

calc_full_scale_time()

Calculate full scale timesteps in [s].

calc_model_mass_flow_rate()

Calculate the model scale flow rate in [kg/s].

calc_net_concentration()

Calculate net concentration in [ppmV].

calc_wtref_mean()

Calculate scaled wtref mean in [m/s].

clear_zeros()

Clear and count zeros in concentration measurements.

convert_temperature()

Convert ambient temperature to °K.

classmethod from_file(*filename*)

Create PointConcentration object from file. open_rate is converted to %.

full_scale_information(*full_scale_wtref*, *full_scale_flow_rate*)

Collect information on desired full scale information. full_scale_wtref in [m/s]. full_scale_flow_rate is automatically adjusted to standard atmosphere conditions. input in [kg/s], output in [m^3/s].

save2file_avg (*filename*, *out_dir=None*)
Save average full scale and model scale data from PointConcentration object to txt file. filename must include '.txt' ending. If no out_dir directory is provided './' is set as standard. @parameter: filename, type = str @parameter: out_dir, type = str

save2file_fs (*filename*, *out_dir=None*)
Save full scale and model scale data from PointConcentration object to txt file. filename must include '.txt' ending. If no out_dir directory is provided './' is set as standard. @parameter: filename, type = str @parameter: out_dir, type = str

save2file_ms (*filename*, *out_dir=None*)
Save model scale data from PointConcentration object to txt file. filename must include '.txt' ending. If no out_dir directory is provided './' is set as standard. @parameter: filename, type = str @parameter: out_dir, type = str

scaling_information (*scaling_factor*, *scale*, *ref_length*, *ref_height*)
Collect data necessary to scale the results. unit: [m], where applicable.

to_full_scale ()
Return all quantities to full scale. Requires XXXXXX to be specified.

tracer_information (*gas_name*, *mol_weight*, *gas_factor*)
Collect information on tracer gas used during measurement. Molecular weight in [kg/mol].

class windtunnel.**PuffConcentration** (*time*, *wtref*, *slow_FID*, *fast_FID*, *signal*, *open_rate*)
PuffConcentration is a class that holds data collected during a puff release point concentration measurement. The class can hold the raw time series, the corresponding wtref and all other quantities necessary to analyse the time series. The PuffConcentration class inherits from pandas.DataFrame, thus offers all of the functionality offered by pandas (e.g. DataFrame.plot.hist(), DataFrame.to_excel(), or DataFrame.rolling().mean()) All the information in a PuffConcentration object can be saved to a txt file, as well as all file type offered by pandas. @parameter: time, type = pd.Series @parameter: wtref, type = np.array @parameter: fast_FID, type = pd.Series @parameter: slow_FID, type = pd.Series @parameter: signal, type = np.array @parameter: open_rate, type = np.array

apply_threshold_concentration (*threshold_concentration=0.0*)
Apply a given threshold concentration to peak_concentration to remove weak puffs. The default value for threshold_concentration is 0. (float).

avg_arrival_time
Get average arrival time.

avg_ascent_time
Get average ascent time.

avg_descent_time
Get average descent time.

avg_leaving_time
Get average leaving time.

avg_peak_concentration
Get average peak concentration.

avg_peak_time
Get average peak time.

calc_net_concentration ()
Calculate net concentration in [ppmV].

calc_release_length ()
Calculate the length of each release period. Returns an np.array containing the duration of each release

period.

check_against_avg_puff()
Check each puff against the average puff of the time series.

detect_arrival_time()
Detects the beginning of each puff. Returns an np.array containing the first timestamp of each puff.

detect_begin_release_index()
Detects the indices of the end of each release period. Returns a list containing the index of the last timestamp of each release period.

detect_begin_release_period()
Detects the beginning of each release period. Returns an np.array containing the first timestamp of each release period.

detect_end_release_index()
Detects the indices of the end of each release period. Returns a list containing the index of the last timestamp of each release period.

detect_end_release_period()
Detects the end of each release period. Returns an np.array containing the last timestamp of each release period.

detect_leaving_time()
Detects the end of each puff. Returns an np.array containing the last timestamp of each puff.

classmethod from_file(*filename*)
Create PuffConcentration object from file. open_rate is converted to %. :type filename: str

get_ascent_time()
Calculate the ascent time between arrrival time and peak time. Returns an np.array.

get_descent_time()
Calculate the ascent time between arrrival time and peak time. Returns an np.array.

get_dosage()
Calculates the dosage of each puff between two release times.

get_peak_concentration()
Acquire peak concentration of each puff. Returns a list.

get_peak_time()
Acquire peak time of each puff. Returns a list.

get_puff_statistics()
Returns DataFrame with all puff information.

get_residence_time()
Calculate the residence time of each puff. Returns an np.array.

max_puffs
Get maximum number of puffs. Deduced from the length of release_length.

offset_correction()
Correct a non-zero offset in the concentration measured.

save2file(*filename*, *out_dir=None*)
Save data from PointConcentration object to txt file. filename must include '.txt' ending. If no out_dir directory is provided './' is set as standard. @parameter: filename, type = str @parameter: out_dir, type = str

class windtunnel.**Timeseries**(*u*, *v*, *x=None*, *y=None*, *z=None*, *t_arr=None*, *t_transit=None*, *tau=10000*)

Timeseries is a class that holds data collected by the BSA software in the standard BSA software output. The class can hold die raw timeseries, the corresponding wtref, the components and coordinates of each measurement as well as the mean wind magnitude and the mean wind direction. The raw timeseries can be processed by nondimensionalising it, adapting the scale, making it equidistant and masking outliers. All the information in a Timeseries object can be saved to a txt file. @parameter: u, type = np.array @parameter: v, type = np.array @parameter: x, type = float @parameter: y, type = float @parameter: z, type = float @parameter: t_arr, type = np.array @parameter: t_transit, type = np.array @parameter: tau, type = int or float - time scale in milliseconds

adapt_scale(*scale*)

Convert timeseries from model scale to full scale. @parameter: scale, type = float

calc_direction()

Calculate wind direction from components.

calc_equidistant_timesteps()

Create equidistant time series.

calc_magnitude()

Calculate wind magnitude from components.

calc_perturbations()

Calculates u' and v' relative to the mean of each tau-long data segment

classmethod from_file(*filename*)

Create Timeseries object from file.

get_wind_comps(*filename*)

Get wind components from filename. @parameter: filename, type = str

get_wtref(*wtref_path*, *filename*, *index=0*, *vscale=1.0*)

Reads wtref-file selected by the time series name 'filename' and scales wtref with vscale. vscale is set to 1 as standard. index accesses only the one wtref value that is associated to the current file. @parameter: path, type = string @parameter: filename, type = string @parameter: index, type = int @parameter: vscale, type = float

mask_outliers(*std_mask=5.0*)

Mask outliers and print number of outliers. std_mask specifies the threshold for a value to be considered an outlier. 5 is the default value for std_mask. @parameter: std_mask, type = float

mean_direction

Calculate mean wind direction from components relative to the wind tunnels axis.

mean_magnitude

Calculate mean wind magnitude from unweighted components.

nondimensionalise()

Nondimensionalise the data. wtref is set to 1 if no wtref is speciefied.

save2file(*filename*, *out_dir=None*)

Save data from Timeseries object to txt file. filename must include '.txt' ending. If no out_dir directory is provided 'C:/Users/[your_u_number]/Desktop/LDA-Analysis/' is set as standard. @parameter: filename, type = str @parameter: out_dir, type = str

set_tau(*milliseconds*)

Give tau a new value

weighted_component_mean

Weigh the u and v component with its transit time through the measurement volume. This is analoguous to the processing of the raw data in the BSA software. Transit time weighting removes a possible bias towards higher wind velocities. Returns the weighted u and v component means.

weighted_component_variance
Weigh the u and v component with its transit time through the measurement volume. This is analoguous to the processing of the raw data in the BSA software. Transit time weighting removes a possible bias towards higher wind velocities. Returns the weighted u and v component variance.

wind_direction_mag_less_180()
Return the wind direction in the range -180 to +180 degrees.

class windtunnel.**Timeseries_nc**(*comp_1*, *comp_2*, *x=None*, *y=None*, *z=None*, *t_arr_1=None*, *t_transit_1=None*, *t_arr_2=None*, *t_transit_2=None*)
Timeseries is a class that holds data collected by the BSA software in non-coincidence mode using the standard BSA software output. The class can hold die raw timeseries, the corresponding wtref, the components and coordinates of each measurement as well as the mean wind magnitude and the mean wind direction. The raw timeseries can be processed by nondimensionalising it, adapting the scale, making it equidistant and masking outliers. All the information in a Timeseries object can be saved to a txt file. @parameter: u, type = np.array @parameter: v, type = np.array @parameter: x, type = float @parameter: y, type = float @parameter: z, type = float @parameter: t_arr, type = np.array @parameter: t_transit, type = np.array

adapt_scale(*scale*)
Convert timeseries from model scale to full scale. @parameter: scale, type = float

calc_direction()
Calculate wind direction from components.

calc_magnitude()
Calculate wind magnitude from components.

equidistant()
Create equidistant time series.

classmethod from_file(*filename*)
Create Timeseries object from file.

get_wind_comps(*filename*)
Get wind components from filename. @parameter: filename, type = str

get_wtref(*wtref_path*, *filename*, *index=0*, *vscale=1.0*)
Reads wtref-file selected by the time series name 'filename' and scales wtref with vscale. vscale is set to 1 as standard. index accesses only the one wtref value that is associated to the current file. @parameter: path, type = string @parameter: filename, type = string @parameter: index, type = int @parameter: vscale, type = float

mask_outliers(*std_mask=5.0*)
Mask outliers and print number of outliers. std_mask specifies the threshold for a value to be considered an outlier. 5 is the default value for std_mask. @parameter: std_mask, type = float

mean_direction
Calculate mean wind direction from components relative to the wind tunnels axis.

mean_magnitude
Calculate mean wind magnitude from unweighted components.

nondimensionalise()
Nondimensionalise the data. wtref is set to 1 if no wtref is speciefied.

pair_components(*atol=1*)
Pair components in comp_1 and comp_2 using atol as absolute tolerance to match a pair of measurements. atol is set to 1 as default, its unit is [ms]. @parameter: atol, type = float or int

save2file(*filename*, *out_dir=None*)
Save data from Timeseries object to txt file. filename must include '.txt' ending. If no out_dir directory is provided './' is set as standard. @parameter: filename, type = str @parameter: out_dir, type = str

weighted_component_mean
Weigh the u and v component with its transit time through the measurement volume. This is analoguous to the processing of the raw data in the BSA software. Transit time weighting removes a possible bias towards higher wind velocities. Returns the weighted u and v component means.

weighted_component_variance
Weigh the u and v component with its transit time through the measurement volume. This is analoguous to the processing of the raw data in the BSA software. Transit time weighting removes a possible bias towards higher wind velocities. Returns the weighted u and v component variance.

`windtunnel.`**adapt_scale**(*x*, *y*, *z*, *t_arr*, *scale*)
Convert timeseries from model scale to full scale. @parameter: x, type = int or float @parameter: y, type = int or float @parameter: z, type = int or float @parameter: t_arr, type = np.array @parameter: scale, type = float

`windtunnel.`**calc_acorr**(*timeseries*, *maxlags*)
Full autocorrelation of time series for lags up to maxlags. @parameter timeseries: np.array or list @parameter maxlags: int

`windtunnel.`**calc_alpha**(*u_mean*, *heights*, *d0=0.0*, *sfc_height=120.0*, *BL_height=600.0*)
Estimate the power law exponent alpha. @parameter: u_mean, type = list or np.array @parameter: heights, type = list or np.array @parameter: d0, type = float @parameter: sfc_height, type = float @parameter: BL_height, type = float

`windtunnel.`**calc_autocorr**(*timeseries*, *lag=1*)
Autocorrelation of time series with lag. @parameter tiemseries: np.array or list @parameter lag: int

`windtunnel.`**calc_exceedance_prob**(*data*, *threshold*)
Calculates exceedance probability of threshold in data. Returns threshold and exceedance probability in percent. @parameter data: @parameter threshold: int

`windtunnel.`**calc_intervalmean**(*indata*, *intervals*, *DD=False*)
Calculates interval means of indata. If DD is set to True the means are calculated for circular quantities. Returns a dictionary with intervals as keys. If intervals has length 1 the function returns an array. @parameter: indata, type = any @parameter: intervals, type = list @parameter: DD, type = boolean

`windtunnel.`**calc_lux_data**(*dt*, *u_comp*)
Calculates the integral length scale according to R. Fischer (2011) from an equidistant time series of the u component using time step dt. @parameter: t_eq, type = int or float @parameter: u_comp, type = np.array or list

`windtunnel.`**calc_lux_data_wght**(*transit_time*, *dt*, *u_comp*)
Calculates the integral length scale according to R. Fischer (2011) from an equidistant time series of the u component using time step dt. @parameter: t_eq, type = int or float @parameter: u_comp, type = np.array or list

`windtunnel.`**calc_ref_spectra**(*reduced_freq*, *a*, *b*, *c*, *d*, *e*)
Calculate dimensionless reference spectra. ??? @parameter: reduced_freq, type = ??? @parameter: a, type = ??? @parameter: b, type = ??? @parameter: c, type = ??? @parameter: d, type = ??? @parameter: e, type = ???

`windtunnel.`**calc_spectra**(*u_comp*, *v_comp*, *t_eq*, *height*)
Calculate dimensionless energy density spectra from an equidistant time series. @parameter: u_comp, type = np.array or list @parameter: v_comp, type = np.array or list @parameter: t_eq, type = np.array or list

`windtunnel.`**calc_stats**(*sets*, *DD=False*)
Returns mean, standard deviation and variance of data in sets. If DD is true then the circular equivalents are calculated. TO BE USED WITH CAUTION @parameter sets: iterable set of data @parameter DD: boolean

`windtunnel.`**calc_turb_data**(*u_comp*, *v_comp*)

Calculate turbulence intensity and turbulent fluxes from equidistant times series of u and v components. @parameter: u_comp: np.array or list @parameter: v_comp: np.array or list

`windtunnel.`**`calc_turb_data_wght`**(*transit_time*, *u_comp*, *v_comp*)

Calculate turbulence intensity and turbulent fluxes from equidistant times series of u and v components using transit time weighted statistics. @parameter: transit_time. type = np.array @parameter: u_compy type = np.array @parameter: v_comp, type = np.array

`windtunnel.`**`calc_wind_stats`**(*u_comp*, *v_comp*, *wdir=0.0*)

Calculate wind data from equidistant times series of u and v components. wdir is a reference wind direction. @parameter: u_comp: np.array or list @parameter: v_comp: np.array or list @parameter: wdir: int

`windtunnel.`**`calc_wind_stats_wght`**(*transit_time*, *u_comp*, *v_comp*, *wdir=0.0*)

Calculate wind data from equidistant times series of u and v components. wdir is a reference wind direction. @parameter: transit_time, type = np.array @parameter: u_comp, type = np.array @parameter: v_comp, type = np.array @parameter: wdir: int

`windtunnel.`**`calc_z0`**(*u_mean*, *heights*, *d0=0.0*, *sfc_height=120.0*, *BL_height=600.0*)

Estimate the roughness length z0. @parameter: u_mean, type = list or np.array @parameter: heights, type = list or np.array @parameter: d0, type = float @parameter: sfc_height, type = float @parameter: BL_height, type = float

`windtunnel.`**`check_directory`**(*directory*)

Checks if directory exists. If directory doesn't exist, it is created. @parameter: directory, type = string

`windtunnel.`**`convergence_test_1`**(*data*, *blocksize=100*)

Conducts a block-wise convergence test on non circular data using blocksize for the size of each increment. Returns a dictionary block_data. Each entry is named after its respective interval. blocksize's default value is 100. @parameter: data, type = np.array or list @parameter: blocksize, type = int or float

`windtunnel.`**`convergence_test_2`**(*data*, *interval=100*, *blocksize=100*)

Conducts a block-wise convergence test on non circular data using blocksize for the size of each increment between intervals. Returns a dictionary block_data. Each entry is named after its respective interval. blocksize's and interval's default values are 100. @parameter: data, type = np.array or list @parameter: interval, type = int @parameter: blocksize, type = int

`windtunnel.`**`count_nan_chunks`**(*data*)

Counts chunks of NaNs in data. Returns the size of each chunk and the overall number of chunks. @parameter: data, type = np.array or string

`windtunnel.`**`equ_dist_ts`**(*arrival_time*, *eq_dist_array*, *data*)

Create a time series with constant time steps. The nearest point of the original time series is used for the corresponding time of the equi-distant time series. @parameter: arrival_time, type = np.array @parameter: eq_dist_array, type = np.array @parameter: data, type = np.array

`windtunnel.`**`equidistant`**(*u*, *v*, *t_arr*)

Create equidistant time series. @parameter: u, type = np.array @parameter: v, type = np.array @parameter: t_arr, type = np.array or list

`windtunnel.`**`find_block`**(*indata*, *length*, *tolerance*)

Finds block of size length in indata. Tolerance allows some leeway. Returns array. @parameter: indata, type = np.array (1D) @parameter: length, type = int @parameter: tolerance, type = int

`windtunnel.`**`find_nearest`**(*array*, *value*)

Finds nearest element of array to value. @parameter: array, np.array @parameter: value, int or float

`windtunnel.`**`from_file`**(*path*, *filename*)

Create array from timeseries in path + file. @parameter: path, string @parameter: filename, string

`windtunnel.`**`get_files`**(*path*, *filename*)
Finds files with filename in path as specified. Filename supports the Unix shell-style wildcards. @parameter: path, type = string @parameter: filename, type = string

`windtunnel.`**`get_lux_referencedata`**(*ref_path=None*)
Reads and returns reference data for the integral length scale (Lux). This function takes no parameters.

`windtunnel.`**`get_pdf_max`**(*data*)
Finds maximum of the probability distribution of data. @parameter data: np.array

`windtunnel.`**`get_percentiles`**(*data_dict*, *percentile_list*)
Get percentiles from each entry in data_dict specified in percentile_list. Returns a dictionary with the results. @parameter: data_dict, type = dict @parameter: percentile_list, type = list

`windtunnel.`**`get_reference_spectra`**(*height*, *ref_path=None*)
Get referemce spectra from pre-defined location.

`windtunnel.`**`get_turb_referencedata`**(*component*, *ref_path=None*)
Reads and returns the VDI reference data for the turbulence intensity of component. @parameter: component, type = string

`windtunnel.`**`get_wind_comps`**(*path*, *filename*)
Get wind components from filename. @parameter: filename, type = str

`windtunnel.`**`get_wtref`**(*wtref_path*, *filename*, *index=0*, *vscale=1.0*)
Reads wtref-file selected by the time series name 'filename' and scales wtref with vscale. vscale is set to 1 as standard. index accesses only the one wtref value that is associated to the current file. @parameter: path, type = string @parameter: filename, type = string @parameter: index, type = int @parameter: vscale, type = float

`windtunnel.`**`mask_outliers`**(*u*, *v*, *std_mask=5.0*)
Mask outliers and print number of outliers. std_mask specifies the threshold for a value to be considered an outlier. 5 is the default value for std_mask. @parameter: u, type = np.array @parameter: v, type = np.array @parameter: std_mask, type = float

`windtunnel.`**`mask_outliers_wght`**(*transit_time*, *u*, *v*, *std_mask=5.0*)
Mask outliers and print number of outliers. std_mask specifies the threshold for a value to be considered an outlier. 5 is the default value for std_mask. This function usues time transit time weighted statistics. @parameter: u, type = np.array @parameter: v, type = np.array @parameter: std_mask, type = float

`windtunnel.`**`nondimensionalise`**(*u*, *v*, *wtref=None*)
Nondimensionalise the data. wtref is set to 1 if no wtref is specified. @parameter: u, type = np.array @parameter: v, type = np.array @parameter: wtref, type = int or float

`windtunnel.`**`plot_DWD_windrose`**(*inFF*, *inDD*)
Plots windrose according to DWD classes of 1 m/s for velocity data and 30 degree classes for directional data. The representation of the windrose in this function is less detailed than in plotwindrose(). @parameter inFF: np.array @parameter inDD: np.array

`windtunnel.`**`plot_JTFA_STFT`**(*u1*, *v1*, *t_eq*, *height*, *second_comp='v'*, *window_length=3500*, *fixed_limits=(None, None)*, *ymax=None*)
Plots the joint time frequency analysis using a short-time Fourier transform smoothed and raw for both wind components in one figure. Returns the figure. To change overlap, @parameter: u1: array of u-component perturbations @parameter: v1: array of second-component perturbations @parameter: t_eq: as defined by Timeseries @parameter: height: z as defined by Timeseries @parameter: second_comp, type = string: the name of the second measured wind component @parameter: window_length, type = int: window length in ms

`windtunnel.`**`plot_Re_independence`**(*data*, *wtref*, *yerr=0*, *ax=None*, ***kwargs*)
Plots the results for a Reynolds Number Independence test from a non- dimensionalised timeseries. yerr specifies the uncertainty. Its default value is 0. @parameter: data, type = np.array or list @parameter: wtref, type = np.array or list @parameter: yerr, type = int or float @parameter: ax: axis passed to function @parameter: kwargs: additional keyword arguments passed to plt.plot()

`windtunnel.`**`plot_boxplots`**(*data_dict*, *ylabel=None*, ***kwargs*)
Plot statistics of concentration measurements in boxplots. Expects input from PointConcentration class. @parameters: data, type = dict @parameters: ylabel, type = string @parameter ax: axis passed to function @parameter kwargs : additional keyword arguments passed to plt.boxplot()

`windtunnel.`**`plot_cdfs`**(*sets*, *lablist*, *ax=None*, ***kwargs*)
Plots CDFs of data in sets using the respective labels from lablist @parameter sets: iterable set of data @parameter lablist: list of strings @parameter ax: axis passed to function @parameter kwargs : additional keyword arguments passed to plt.plot()

`windtunnel.`**`plot_convergence`**(*data_dict*, *ncols=3*, ***kwargs*)
Plots results of convergence tests performed on any number of quantities in one plot. ncols specifies the number of columns desired in the output plot. kwargs contains any parameters to be passed to plot_convergence_test, such as wtref, ref_length and scale. See doc_string of plot_convergence_test for more details. @parameter: data_dict, type = dictionary @parameter: ncols, type = int @parameter: kwargs keyword arguments passed to plot_convergence_test

`windtunnel.`**`plot_convergence_test`**(*data*, *wtref=1*, *ref_length=1*, *scale=1*, *ylabel=”*, *ax=None*, ***kwargs*)
Plots results of convergence tests from data. This is a very limited function and is only intended to give a brief overview of the convergence rest results using dictionaries as input objects. wtref, ref_length and scale are used to determine a dimensionless time unit on the x-axis. Default values for each are 1. @parameter: data_dict, type = dictionary @parameter: wtref, type = float or int @parameter: ref_length, type = float or int @parameter: scale, type = float or int @parameter: ylabel, type = string @parameter: ax: axis passed to function

`windtunnel.`**`plot_fluxes`**(*data*, *heights*, *yerr=0*, *component='v'*, *lat=False*, *ax=None*, ***kwargs*)
Plots fluxes from data for their respective height with a 10% range of the low point mean. yerr specifies the uncertainty. Its default value is 0. WARNING: Data must be made dimensionless before plotting! If lat is True then a lateral profile is created. @parameter: data, type = list or np.array @parameter: height, type = list or np.array @parameter: yerr, type = int or float @parameter: component, type = string @parameter: lat, type = boolean @parameter ax: axis passed to function @parameter kwargs : additional keyword arguments passed to plt.plot()

`windtunnel.`**`plot_fluxes_log`**(*data*, *heights*, *yerr=0*, *component='v'*, *ax=None*, ***kwargs*)
Plots fluxes from data for their respective height on a log scale with a 10% range of the low point mean. yerr specifies the uncertainty. Its default value is 0. WARNING: Data must be made dimensionless before plotting! @parameter: data, type = list or np.array @parameter: height, type = list or np.array @parameter: yerr, type = int or float @parameter: component, type = string @parameter ax: axis passed to function @parameter kwargs : additional keyword arguments passed to plt.plot()

`windtunnel.`**`plot_hist`**(*data*, *ax=None*, ***kwargs*)
Creates a scatter plot of x and y. @parameter: data, type = list or np.array @parameter ax: axis passed to function @parameter kwargs : additional keyword arguments passed to plt.plot()

`windtunnel.`**`plot_lux`**(*Lux*, *heights*, *err=0*, *lat=False*, *ref_path=None*, *ax=None*, ***kwargs*)
Plots Lux data on a double logarithmic scale with reference data. yerr specifies the uncertainty. Its default value is 0. If lat is True then a lateral profile, without a loglog scale, is created. @parameter: Lux, type = list or np.array @parameter: heights, type = list or np.array @parameter: err, type = int or float @parameter: lat, type = boolean @parameter: ref_path = string @parameter ax: axis passed to function @parameter kwargs : additional keyword arguments passed to plt.plot()

`windtunnel.`**`plot_pdfs`**(*sets*, *lablist*, *ax=None*, ***kwargs*)
Plots PDFs of data in sets using the respective labels from lablist. @parameter sets: iterable set of data @parameter lablist: list of strings @parameter ax: axis passed to function @parameter kwargs : additional keyword arguments passed to plt.plot()

`windtunnel.`**`plot_pdfs_err`**(*sets*, *lablist*, *error*, *ax=None*, ***kwargs*)
Plots PDFs of data in sets using the respective labels from lablist with a given margin of error. @parameter sets:

iterable set of data @parameter lablist: list of strings @parameter error: int or float @parameter ax: axis passed to function @parameter kwargs : additional keyword arguments passed to plt.plot()

`windtunnel.`**`plot_perturbation_rose`**(*u1*, *v1*, *total_mag*, *total_direction*, *bar_divider=3000*, *second_comp='v'*)

Plots a detailed wind rose using only the perturbation component of the wind. Number of bars depends on bar_divider and length of u1. @parameter: u1: array of u-component perturbations @parameter: v1: array of second-component perturbations @parameter: total_mag: array containing magnitude of wind (not perturbation) @parameter: total_direction: array containing direction of wind (not perturbation) @parameter: bar_divider: inversely proportional to number of bars to be plotted @parameter: second_comp, type = string: the name of the second measured wind component

`windtunnel.`**`plot_rose`**(*inFF*, *inDD*, *ff_steps*, *dd_range*)

Plots windrose according to user specified input from ff_steps and dd_Range. @parameter: inFF, type = np.array @parameter: inDD, type = np.array @parameter: ff_steps, type = list or np.array @parameter: dd_range, type = int or float

`windtunnel.`**`plot_scatter`**(*x*, *y*, *std_mask=5.0*, *ax=None*, ***kwargs*)

Creates a scatter plot of x and y. All outliers outside of 5 STDs of the components mean value are coloured in orange. @parameter: x, type = list or np.array @parameter: y, type = list or np.array @parameter: std_mask, float @parameter ax: axis passed to function @parameter kwargs : additional keyword arguments passed to plt.scatter()

`windtunnel.`**`plot_spectra`**(*f_sm*, *S_uu_sm*, *S_vv_sm*, *S_uv_sm*, *u_aliasing*, *v_aliasing*, *uv_aliasing*, *wind_comps*, *height*, *ref_path=None*, *ax=None*, ***kwargs*)

Plots spectra using INPUT with reference data. @parameter: ??? @parameter: ref_path, type = string @parameter ax: axis passed to function @parameter kwargs : additional keyword arguments passed to plt.plot()

`windtunnel.`**`plot_stdevs`**(*data*, *t_eq*, *tau*, *comp='u'*, *ax=None*, ***kwargs*)

This function plots the spread of an array based on how many standard deviations each point is from the mean over each tau-long time period @parameter: data, type = np.array (the array to be analysed) @parameter: t_eq, type = np.array (corresponding times steps in [ms]) @parameter: tau, type = int or float (characteristic time scale (ms) @parameter: ax, axis passed to function @parameter kwargs : additional keyword arguments passed to ax.bar()

`windtunnel.`**`plot_turb_int`**(*data*, *heights*, *yerr=0*, *component='I_u'*, *lat=False*, *ref_path=None*, *ax=None*, ***kwargs*)

Plots turbulence intensities from data with VDI reference data for their respective height. yerr specifies the uncertainty. Its default value is 0. If lat is True then a lateral profile is created. @parameter: data, type = list or np.array @parameter: heights, type = list or np.array @parameter: yerr, type = int or float @parameter: component, type = string @parameter: lat, type = boolean @parameter: ref_path, type = string @parameter: ax, axis passed to function @parameter kwargs : additional keyword arguments passed to plt.plot()

`windtunnel.`**`plot_winddata`**(*mean_magnitude*, *u_mean*, *v_mean*, *heights*, *yerr=0*, *lat=False*, *ax=None*, ***kwargs*)

Plots wind components and wind magnitude for their respective height. yerr specifies the uncertainty. Its default value is 0. If lat is True then a lateral profile is created. @parameter: mean_magnitude, type = list or np.array @parameter: u_mean, type = list or np.array @parameter: v_mean, type = list or np.array @parameter: heights, type = list or np.array @parameter: yerr, type = int or float @parameter: lat, type = boolean @parameter ax: axis passed to function @parameter kwargs : additional keyword arguments passed to plt.plot()

`windtunnel.`**`plot_winddata_log`**(*mean_magnitude*, *u_mean*, *v_mean*, *heights*, *yerr=0*, *ax=None*, ***kwargs*)

Plots wind components and wind magnitude for their respective height on a log scale. yerr specifies the uncertainty. Its default value is 0. @parameter: mean_magnitude, type = list or np.array @parameter: u_mean, type = list or np.array @parameter: v_mean, type = list or np.array @parameter: heights, type = list or np.array @parameter: yerr, type = int or float @parameter: ax, axis passed to function @parameter kwargs : additional keyword arguments passed to plt.plot()

`windtunnel.`**`plot_windrose`**(*inFF*, *inDD*, *num_bars=10*, *ax=None*, *left_legend=False*)
Plots windrose with dynamic velocity classes of each 10% percentile and 10 degree classes for directional data. The representation of the windrose in this function is more detailed than in plot_DWD_windrose(). @parameter inFF: np.array @parameter inDD: np.array @parameter num_bars: how many segments the degree range should be broken into @parameter ax: pyplot axes object, must be polar @left_legend: if true, the legend is positioned to the left of the plot instead of the right

`windtunnel.`**`power_law`**(*u_comp*, *height*, *u_ref*, *z_ref*, *alpha*, *d0=0*)
Estimate power law profile. @parameter: u_comp, type = int or float @parameter: height, type = int or float @parameter: u_ref, type = int or float @parameter: z_ref, type = int or float @parameter: alpha, type = int or float @parameter: d0, type = int or float

`windtunnel.`**`transit_time_weighted_flux`**(*transit_time*, *component_1*, *component_2*)
Calculate mean flux using transit time weighted statistics. Transit time weighting removes a possible bias towards higher wind velocities. Returns a mean weighted flux. @parameter: transit_time, type = np.arrray([]) @parameter: component_1, type = np.arrray([]) @parameter: component_2, type = np.arrray([])

`windtunnel.`**`transit_time_weighted_mean`**(*transit_time*, *component*)
Weigh the flow component with its transit time through the measurement volume. This is analoguous to the processing of the raw data in the BSA software. Transit time weighting removes a possible bias towards higher wind velocities. Returns the weighted component mean. @parameter: transit_time, type = np.arrray([]) @parameter: component, type = np.arrray([])

`windtunnel.`**`transit_time_weighted_var`**(*transit_time*, *component*)
Weigh the u and v component with its transit time through the measurement volume. This is analoguous to the processing of the raw data in the BSA software. Transit time weighting removes a possible bias towards higher wind velocities. Returns the weighted u and v component variance. @parameter: transit_time, type = np.arrray([]) @parameter: component, type = np.arrray([])

`windtunnel.`**`trunc_at`**(*string*, *delimiter*, *n=3*)
Returns string truncated at the n'th (3rd by default) occurrence of the delimiter.

CHAPTER FIVE

INDEX

- genindex

PYTHON MODULE INDEX

W

INDEX

E

F

G

M

N

O

P

S

T

W