PyGSL Reference Manual

Release 0.9

Achim Gädke
Jochen Küpper
Pierre Schnizer

October, 2008

Technische Universität Darmstadt, Darmstadt, Germany
achimgaedke@users.sourceforge.net
Gesellschaft für Schwerionenforschung, Darmstadt, Germany
schnizer@users.sourceforge.net

Abstract

pygsl grants python users access to the GNU scientific library. The latest version can be found at the project homepage, `http://pygsl.sf.net`.

Implemented features:

`pygsl.blas`	basic linear algebra system
`pygsl.chebyshev`	chebyshev approximations
`pygsl.combination`	combinations
`pygsl.const`	$>$ 200 often used mathematical and scientific constants.
`pygsl.diff`	(Deprecated. Use pygsl.deriv instead).
`pygsl.deriv`	Numerical differentiation.
`pygsl.eigen`	
`pygsl.fit`	
`pygsl.histogram`	1d and 2d histograms and operations on histograms.
`pygsl.ieee`	Access to the ieee-arithmetics layer of gsl.
`pygsl.integrate`	
`pygsl.interpolation`	
`pygsl.linalg`	
`pygsl.math`	
`pygsl.monte`	
`pygsl.minimize`	
`pygsl.multifit`	
`pygsl.multifit_nlin`	
`pygsl.multimin`	
`pygsl.multiroots`	
`pygsl.odeiv`	
`pygsl.permutation`	
`pygsl.poly`	
`pygsl.qrng`	
`pygsl.rng`	random number generators and probability densities.
`pygsl.roots`	
`pygsl.siman`	Simulated anealing
`pygsl.sum`	
`pygsl.sf`	$>$ 200 special functions.
`pygsl.statistics`	Statistical functions.

CONTENTS

CHAPTER

ONE

System Requirements, Installation

1.1 Status

Status of GSL-Library The gsl-library is since version 1.0 stable and for general use. More information about it at `http://www.gnu.org/software/gsl/`.

Status of this interface Nearly all modules are wrapped. A lot of tests are covering various functionality. Please report to the mailing list `pygsl-discuss@lists.sourceforge.net` if you find a bug.

The hankel modules have been wrapped. Please write to the mailing list `pygsl-discuss@lists.sourceforge.net` if you require one of the modules and are willing to help with a simple example. If any other function is missing or some other module (e.g. ntuple) or function, do not hesitate to write to the list.

Retriving the Interface You can download it here: `http://sourceforge.net/projects/pygsl`

1.2 Requirements

To build the interface, you will need

- gsl-1.x,
- python2.2 or better,
- NumPy, and
- a c compiler (like gcc).

Supported Platforms are:

- Linux (Redhat/Debian/SuSE) with python2.* and gsl-1.*
- Win32

It was tested and is tested on an irregular basis on the following platforms

- SUN
- Cygwin
- MacOS X

but is supposed to build on any POSIX platforms.

1.3 Installing the pygsl interface

gsl-config must be on your path:

```
# unpack the source distribution
gzip -d -c pygsl-x.y.z.tar.gz|tar xvf-
cd pygsl-x.y.z
# do this with your prefered python version
# to set the gsl location explicitly use setup.py --gsl-prefix=/path/to/gsl
python setup.py build
# change to an user id, that is allowed to do installation
python setup.py install
```

Ready....

Do not test the interface in the distribution root or in the directories 'src' or 'pygsl'.

1.3.1 Building on win32

Windows by default does not allow to run a posix shell. Here a different path is required. First change into the directory 'gsl_dist'. Copy the file 'gsl_site_example.py' and edit it to reflect your installation of GSL and SWIG if you want to run it yourself. The pygsl windows binaries distributed over `http://sourceforge.net/projects/pygsl/` are built using the mingw32 compiler.

Uninstall GSL interface `rm -r` "python install path"`/lib/python`"version"`/site-packages/pygsl`

Testing the directory 'tests' contains several testsuites, based on python `unittest`. The script 'run_test.py' in this directory will run one after the other.

Support Please send mails to our mailinglist at pygsl-discuss@lists.sourceforge.net.

Developement You can browse our cvs tree at `http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/pygsl/pygsl/`.
Type this to check out the actual version:

```
cvs -d:pserver:anonymous@cvs.pygsl.sourceforge.net:/cvsroot/pygsl login
#Hit return for no password.
cvs -z3 -d:pserver:anonymous@cvs.pygsl.sourceforge.net:/cvsroot/pygsl co pygsl
```

The script **tools/extract_tool.py** generates most of the special function code.

ToDo Implement other parts:

History

- a gsl-interface for python was needed for a project at Center for Applied Informatics Cologne.
- 'gsl-0.0.3' was released at May 23, 2001
- 'gsl-0.0.4' was released at January 8, 2002

- ‘gsl-0.0.5’ is growing since January, 2002
- ‘gsl-0.2.0’ was released at
- ‘gsl-0.3.0’ was released at
- ‘gsl-0.3.1’ was released at
- ‘gsl-0.3.2’ was released at
- ‘gsl-0.9.4’ was released at 25. October 2008

Thanks Jochen Küpper (jochen@jochen-kuepper.de) for `pygsl.statistics` part
Fabian Jakobs for `pygsl.blas`, `pygsl.eigen` `pygsl.linalg`, `pygsl.permutation`
Leonardo Milano for rpm build
Eric Gurrola and Peter Stoltz for testing and supporting the port of pygsl to the MAC
Sebastien Maret for supporting the Fink `http://fink.sourceforge.net` port of pygsl.

Maintainers Achim Gädke (AchimGaedke@users.sourceforge.net),
Pierre Schnizer (schnizer@users.sourceforge.net)

1.4 Information for Distributors

The information given here is intended for people distributing pygsl as well as for people who are not afraid to run code generation tools in case building pygsl fails.

1.4.1 Preferable array module

The numpy array module should be the first choice these days.

1.4.2 Wrapper generated modules

The following modules are generated by SWIG wrappers:

- _block
- _callback
- gslwrap
- hankel
- _poly
- bspline

In case on of these modules shows a compilation error, just delete the corresponding file in the swig_src directory (e.g. swig_src/poly_wrap.c if for the poly module.). Set the variable `USE_SWIG` to `1` in the ‘setup.py’ file. Run the build process again. Now ‘setup.py’ should call swig and rebuild the module wrappers.

1.4.3 Dedicated wrappers

Some modules come with their own wrapping tools which are implemented in python.

1.4.4 The const module

Change into the src directory. Define the shell variable `PYGSL_GSLCVS` variable, so that it either points to the GSL CVS repository or to the include directory of your GSL installation (typically /usr/include on a linux installation). Then type `python ../tools/constants_tool.py` This will generate the files 'const_m_array.c' 'const_num_array.c' 'const_cgsm_array.c' 'const_mksa_array.c'

1.4.5 The special function module.

PyGSL comes with two implementations for the special functions. The sf and the testing.sf module. The first one was the first implementation and will be removed soon, while the later one wraps the special functions as *numpy* UFuncs. The UFunc module will eventually replace the sf module.

Generating the wrapper has not been automatized yet. First swig is used to parse the header files and dump them into xml and then a specialized wrapper reads this tree and writes the wrappers.

This involves the following steps:

- Change into the directory 'testing/tools'
- Run `swig -python -I<gsl header directory> -xml swig_test.i` So on a linux machine this is typically `swig -python -I/usr/include -xml swig_test.i`
- Change into the directory 'testing/src/sf'
- Run the wrapper generator using `python ../../tools/sf_functions.py`
- Run the build process. Set `BUILD_TESTING = 1` in the setup.py file. If required update the time stamp of the file 'testing/src/sf/sfmodule_testing.c'. This will build the module `_ufuncs`.

The functions not wrapped automatically are defined in the list `exclude_list` in the file 'sf_functions.py'

CHAPTER

TWO

Design of the PyGSL interface

The GSL library was implemented using the C language. This implies that each function uses a certain type for the different variables and are fixed to one specific type. The wrapper will try to convert each argument to the approbriate C type. The PyGSL interface tries to follow it as much as possible but only as far as resonable. For example the definition of the poly_eval function in C is given by
double gsl_poly_eval(*const double C[], const int LEN, const double X*)
The corresponding python wrapper was implemented by
poly.poly_eval(*C, x*)
as the wrapper can get the length of any python object and then fill the len variable. The mathematical calculation is performed by the GSL library. Thus the calculation is limited to the precision provided by the underlying hardware.

Default arguments are used to allocate workspaces on the fly if not provided by the user. Consider for example the fft module. The function for the real forward transform is named

int gsl_fft_real_transform(*double DATA[], size_t STRIDE, size_t N,*
*const gsl_fft_real_wavetable * WAVETABLE,*
*gsl_fft_real_workspace * WORK*)

The corresponding python wrapper is found in the fft module called real_transform
real_transform(*data,* [*space, table, output*])
The wrapper will get the stride and size information from the data object provided by the user. If space or table are not provided, these objects will be generated on the fly. As the GSL function applies the transformation in space, an internal copy is made of the data and only then the object is passed to the GSL function. If an output object is provided the data will be copied there instead. PyGSL will always make copies of objects which would be otherwise modified in place.

2.1 Callbacks

Solvers require as one argument a user function to work on which have to be provided by the user. These callbacks typically are of the form
f (*x, params*)
    ...
    return result
Please note that this function must return the exact number of arguments as given in the example. The wrappers around callbacks go a long way to try to provide meaningfull error messages. If a solver fails, please check that the number of input and output arguments it takes are correct

2.2 Error handling

As GSL is a C library error handling is implemented using an error handler and return values. PyGSL generates python exceptions out of these values. See `pygsl.errors` (chapter 3) for a list of the exceptions.

2.3 Exception handling

GSL provides a selectable error handler, that is called for occuring errors (like domain errors, division by zero, etc.). This is switched off. Instead each wrapper function will check the error return value and in case of error an python exception is created.

Here is a python level example:

```
import pygsl.histogram
import pygsl.errors
hist=pygsl.histogram.histogram2d(100,100)
try:
   hist[-1,-1]=0
except pygsl.errors.gsl_Error,err:
   print err
```

Will result

```
input domain error: index i lies outside valid range of 0 .. nx - 1
```

An exception are ufuncs in the testings.sf module (see section12.2).

2.3.1 Change of internal error handling.

Before a error handler was installed by init_pygsl into gsl which translated the error code (and the message) to a python exception. This required that the GIL was available, which numpy ufuncs dispose. Thus now this gsl error handler is deactivated and instead the C error code returned by the C function is translated to an error code by the wrapper called from python.

UFuncs do not call this handler now at all.

2.4 The documentation gap

PyGSL does still lack an approbriate documentation. Most documentation is accessible over the internal documentation strings. These are accessible as `__doc__` attributes (the help function does not always show them). It can be sometimes necessary to create an object to see its methods as well as the documentation of the methods (e.g.a random number generator in the rng module to see its methods). The 'example' directory contains examples for (nearly each) module.

Please feel welcome to add to the documentation!

Acknowledgment Parts of this this manual are based on the GNU Scientific Library reference manual. The authors want to thank all for contribution of code, support material for generating distribution packages, bug reports and example scripts.

CHAPTER

THREE

pygsl.errors
Error and warning classes

This chapter provides information about the gsl_Error exception class that comes with this module.

3.1 Exception Classes

exception gsl_Error()
derived from Exception, can be constructed with any object as parameter. It is baseclass to all other GSL Exceptions

These classes are translations of the '¡gsl/gsl_errno.h¿' to python exceptions.

exception gsl_ArithmeticError()
derived from gsl_Error and exceptions.ArithmeticError, base of all common arithmetic exceptions

exception gsl_OverflowError()
derived from gsl_Error and exceptions.OverflowError

exception gsl_ZeroDivisionError()
derived from gsl_Error and exceptions.ZeroDivisionError

exception gsl_FloatingPointError()
derived from gsl_Error and exceptions.FloatingPointError

exception gsl_ArithmeticError()
is derived from gsl_Error and from ArithmeticError . This exception is the base of all common arithmetic exceptions.

exception gsl_AccuracyLossError()
is derived from gsl_ArithmeticError . This exception is raised if the failed to reach the specified tolerance.

exception gsl_BadFuncError()
is derived from gsl_Error . This exception is raised if problem with a user-supplied function occur.

exception gsl_BadLength()
is derived from gsl_Error . This exception is raised if matrix or vector lengths are not conformant.

exception gsl_BadToleranceError()
is derived from gsl_Error . This exception is raised if user specified an tolerance which can not be reached.

exception gsl_CacheLimitError()
is derived from gsl_Error . This exception is raised if the cache limit is exceeded.

exception gsl_DivergeError()
is derived from gsl_ArithmeticError . This exception is raised if an integral or series is divergent.

exception gsl_DomainError()
is derived from gsl_Error . This exception is raised if domain errors occure. e.g. sqrt(-1).

exception gsl_EOFError()
is derived from gsl_Error and from EOFError . This exception is raised if end of file .

exception gsl_FactorizationError()
is derived from gsl_Error . This exception is raised if factorization failed.

exception gsl_FloatingPointError()
is derived from gsl_Error and from FloatingPointError .

exception gsl_GenericError()
is derived from gsl_Error .

exception gsl_InvalidArgumentError()
is derived from gsl_Error . This exception is raised if an invalid argument is supplied by the user.

exception gsl_JacobianEvaluationError()
is derived from gsl_ArithmeticError . This exception is raised if jacobian evaluations are not improving the solution.

exception gsl_MatrixNotSquare()
is derived from gsl_Error . This exception is raised if the given matrix is not square.

exception gsl_MaximumIterationError()
is derived from gsl_ArithmeticError . This exception is raised if the maximum number of iterations is exceeded.

exception gsl_NoHardwareSupportError()
is derived from gsl_Error . This exception is raised if the requested feature is not supported by the hardware.

exception gsl_NoProgressError()
is derived from gsl_ArithmeticError . This exception is raised if the iteration is not making progress towards solution.

exception gsl_NotImplementedError()
is derived from gsl_Error and from NotImplementedError . This exception is raised if a requested feature is not (yet) implemented .

exception gsl_OverflowError()
is derived from gsl_Error and from OverflowError .

exception gsl_PointerError()
is derived from gsl_Error . This exception is raised if an invalid pointer is found by the C wrapper code or by the GSL library.

exception gsl_RangeError()
is derived from gsl_ArithmeticError . This exception is raised if output would be out or range, e.g. exp(1e100) .

exception gsl_RoundOffError()
is derived from gsl_ArithmeticError . This exception is raised if arithmetic failed because of roundoff error.

exception gsl_RunAwayError()
is derived from gsl_ArithmeticError . This exception is raised if iterative process is out of control.

exception gsl_SanityCheckError()
is derived from gsl_Error . This exception is raised if a sanity check failed - shouldn't happen.

exception gsl_SingularityError()
is derived from gsl_ArithmeticError . This exception is raised if an apparent singularity is detected.

exception gsl_TableLimitError()
is derived from gsl_Error . This exception is raised if the table limit is exceeded.

exception gsl_ToleranceError()
is derived from gsl_ArithmeticError . This exception is raised if the alghorithm failed to reach the specified tolerance.

exception gsl_ToleranceFError()
is derived from gsl_ArithmeticError . This exception is raised if the alghorithm cannot reach the specified tolerance in F (typically the variation of the evaluated function).

exception gsl_ToleranceGradientError()
is derived from gsl_ArithmeticError . This exception is raised if cannot reach the specified tolerance for the gradient.

exception gsl_ToleranceXError()
is derived from gsl_ArithmeticError . This exception is raised if cannot reach the specified tolerance in X (typically a search result).

exception gsl_UnderflowError()
is derived from gsl_Error and from OverflowError .

exception gsl_ZeroDivisionError()
is derived from gsl_Error and from ZeroDivisionError .

All the above errors are just translations of the errno to python exceptions.

The following two are specific to pygsl:

exception pygsl.errors.pygsl_NotImplementedError()
is derived from gsl_Error and from NotImplementedError . This exception is raised if a feature is requested but not implemented. Currently only used if a module requests the debugging enviroment of the init module, but the init module was not compiled with #define DEBUG=1

exception pygsl.errors.pygsl_StrideError()
is derived from gsl_SanityCheckError . GSL uses as strides multiples of the basis type; for a vector or doubles, one means from one double to the next. Numpy or numarray count the stride in multiples of the size of a char. Therefore the stride has to be recalculated before the approbriate GSL function can be called. If that fails this exception is raised.

3.2 Warning Classes

exception gsl_Warning()
The dedicated warning class for GSL has Warning as base class.

exception gsl_DomainWarning()
derived from gsl_Warning, used by some pygsl.histogram functions

CHAPTER

FOUR

pygsl.const
Mathematical and scientific constants

In this module some usefull constants are defined. There are four groups of constants:

- mathematical,
- physical in mks unit system,
- physical in cgs unit system and
- physical number constants (e.g. fine structure)

The other modules are created during the initialisation of `pygsl.const`. For convenience the mathematical, physical mks constants and number constants also are available in the namespace of `pygsl.const`. If the used GSL version is before gsl1.4, see

```
pygsl.compiled_gsl_version
```

than the module names are cgs and mks. Form gsl1.5 these modules have been renamed to cgsm and mksa. So to use cgs constants one has to write

```
import pygsl.const
import pygsl.const.cgs
print pygsl.const.cgs.speed_of_light/pygsl.const.speed_of_light
```

for gsl $<$ 1.5 and

```
import pygsl.const                                                    .
import pygsl.const.cgsm
print pygsl.const.cgsm.speed_of_light/pygsl.const.speed_of_light
```

Of course the result is `100.0`. Short examples are given at top of each section.

See Also:

The actual values are taken form the GSL headers. The GNU Scientific Library reference provides a more detailed description of these constants.

4.1 pygsl.const.m
Mathematical constants

```
from pygsl.const.m import *
print sqrt2*sqrt2
```

Prints `2.0`.
Here comes the list:

Name	GSL Name	value
e	M_E	e
log2e	M_LOG2E	$\log 2e$
log10e	M_LOG10E	$\log 10e$
sqrt2	M_SQRT2	$\sqrt{2}$
sqrt1_2	M_SQRT1_2	$\sqrt{1/2}$
sqrt3	M_SQRT3	$\sqrt{3}$
pi	M_PI	π
pi_2	M_PI_2	$\pi/2$
pi_4	M_PI_4	$\pi/4$
sqrtpi	M_SQRTPI	$\sqrt{\pi}$
2_sqrtpi	M_2_SQRTPI	$2/\sqrt{\pi}$
1_pi	M_1_PI	$1/\pi$
2_pi	M_2_PI	$2/\pi$
ln10	M_LN10	$\ln 10$
ln2	M_LN2	$\ln 2$
lnpi	M_LNPI	$\ln \pi$
euler	M_EULER	Euler constant

4.2 pygsl.const.mksa
Scientific constants in mksa units

```
from pygsl.const import cgsm
print "a teaspoon contains %g m^3"%mks.teaspoon
```

These are the provided constants:

Name	gsl Name	unit
speed_of_light	GSL_CONST_MKSA_SPEED_OF_LIGHT	m / s
gravitational_constant	GSL_CONST_MKSA_GRAVITATIONAL_CONSTANT	m^3 / kg s^2
plancks_constant_h	GSL_CONST_MKSA_PLANCKS_CONSTANT_H	kg m^2 / s
plancks_constant_hbar	GSL_CONST_MKSA_PLANCKS_CONSTANT_HBAR	kg m^2 / s
vacuum_permeability	GSL_CONST_MKSA_VACUUM_PERMEABILITY	kg m / A^2 s^2
astronomical_unit	GSL_CONST_MKSA_ASTRONOMICAL_UNIT	m
light_year	GSL_CONST_MKSA_LIGHT_YEAR	m
parsec	GSL_CONST_MKSA_PARSEC	m
grav_accel	GSL_CONST_MKSA_GRAV_ACCEL	m / s^2

Name	gsl Name	unit
electron_volt	GSL_CONST_MKSA_ELECTRON_VOLT	kg m^2 / s^2
mass_electron	GSL_CONST_MKSA_MASS_ELECTRON	kg
mass_muon	GSL_CONST_MKSA_MASS_MUON	kg
mass_proton	GSL_CONST_MKSA_MASS_PROTON	kg
mass_neutron	GSL_CONST_MKSA_MASS_NEUTRON	kg
rydberg	GSL_CONST_MKSA_RYDBERG	kg m^2 / s^2
boltzmann	GSL_CONST_MKSA_BOLTZMANN	kg m^2 / K s^2
bohr_magneton	GSL_CONST_MKSA_BOHR_MAGNETON	A m^2
nuclear_magneton	GSL_CONST_MKSA_NUCLEAR_MAGNETON	A m^2
electron_magnetic_moment	GSL_CONST_MKSA_ELECTRON_MAGNETIC_MOMENT	A m^2
proton_magnetic_moment	GSL_CONST_MKSA_PROTON_MAGNETIC_MOMENT	A m^2
molar_gas	GSL_CONST_MKSA_MOLAR_GAS	kg m^2 / K mol s^2
standard_gas_volume	GSL_CONST_MKSA_STANDARD_GAS_VOLUME	m^3 / mol
minute	GSL_CONST_MKSA_MINUTE	s
hour	GSL_CONST_MKSA_HOUR	s
day	GSL_CONST_MKSA_DAY	s
week	GSL_CONST_MKSA_WEEK	s
inch	GSL_CONST_MKSA_INCH	m
foot	GSL_CONST_MKSA_FOOT	m
yard	GSL_CONST_MKSA_YARD	m
mile	GSL_CONST_MKSA_MILE	m
nautical_mile	GSL_CONST_MKSA_NAUTICAL_MILE	m
fathom	GSL_CONST_MKSA_FATHOM	m
mil	GSL_CONST_MKSA_MIL	m
point	GSL_CONST_MKSA_POINT	m
texpoint	GSL_CONST_MKSA_TEXPOINT	m
micron	GSL_CONST_MKSA_MICRON	m
angstrom	GSL_CONST_MKSA_ANGSTROM	m
hectare	GSL_CONST_MKSA_HECTARE	m^2
acre	GSL_CONST_MKSA_ACRE	m^2
barn	GSL_CONST_MKSA_BARN	m^2
liter	GSL_CONST_MKSA_LITER	m^3
us_gallon	GSL_CONST_MKSA_US_GALLON	m^3
quart	GSL_CONST_MKSA_QUART	m^3
pint	GSL_CONST_MKSA_PINT	m^3
cup	GSL_CONST_MKSA_CUP	m^3
fluid_ounce	GSL_CONST_MKSA_FLUID_OUNCE	m^3
tablespoon	GSL_CONST_MKSA_TABLESPOON	m^3
teaspoon	GSL_CONST_MKSA_TEASPOON	m^3
canadian_gallon	GSL_CONST_MKSA_CANADIAN_GALLON	m^3
uk_gallon	GSL_CONST_MKSA_UK_GALLON	m^3
miles_per_hour	GSL_CONST_MKSA_MILES_PER_HOUR	m / s
kilometers_per_hour	GSL_CONST_MKSA_KILOMETERS_PER_HOUR	m / s
knot	GSL_CONST_MKSA_KNOT	m / s
pound_mass	GSL_CONST_MKSA_POUND_MASS	kg
ounce_mass	GSL_CONST_MKSA_OUNCE_MASS	kg
ton	GSL_CONST_MKSA_TON	kg
metric_ton	GSL_CONST_MKSA_METRIC_TON	kg
uk_ton	GSL_CONST_MKSA_UK_TON	kg
troy_ounce	GSL_CONST_MKSA_TROY_OUNCE	kg
carat	GSL_CONST_MKSA_CARAT	kg
unified_atomic_mass	GSL_CONST_MKSA_UNIFIED_ATOMIC_MASS	kg
gram_force	GSL_CONST_MKSA_GRAM_FORCE	kg m / s^2

Name	gsl Name	unit
pound_force	GSL_CONST_MKSA_POUND_FORCE	kg m / s^2
kilopound_force	GSL_CONST_MKSA_KILOPOUND_FORCE	kg m / s^2
poundal	GSL_CONST_MKSA_POUNDAL	kg m / s^2
calorie	GSL_CONST_MKSA_CALORIE	kg m^2 / s^2
btu	GSL_CONST_MKSA_BTU	kg m^2 / s^2
therm	GSL_CONST_MKSA_THERM	kg m^2 / s^2
horsepower	GSL_CONST_MKSA_HORSEPOWER	kg m^2 / s^3
bar	GSL_CONST_MKSA_BAR	kg / m s^2
std_atmosphere	GSL_CONST_MKSA_STD_ATMOSPHERE	kg / m s^2
torr	GSL_CONST_MKSA_TORR	kg / m s^2
meter_of_mercury	GSL_CONST_MKSA_METER_OF_MERCURY	kg / m s^2
inch_of_mercury	GSL_CONST_MKSA_INCH_OF_MERCURY	kg / m s^2
inch_of_water	GSL_CONST_MKSA_INCH_OF_WATER	kg / m s^2
psi	GSL_CONST_MKSA_PSI	kg / m s^2
poise	GSL_CONST_MKSA_POISE	kg / m / s
stokes	GSL_CONST_MKSA_STOKES	m^2 / s
faraday	GSL_CONST_MKSA_FARADAY	A s / mol
electron_charge	GSL_CONST_MKSA_ELECTRON_CHARGE	A s
gauss	GSL_CONST_MKSA_GAUSS	kg / A s^2
stilb	GSL_CONST_MKSA_STILB	cd / m^2
lumen	GSL_CONST_MKSA_LUMEN	cd sr
lux	GSL_CONST_MKSA_LUX	cd sr / m^2
phot	GSL_CONST_MKSA_PHOT	cd sr / m^2
footcandle	GSL_CONST_MKSA_FOOTCANDLE	cd sr / m^2
lambert	GSL_CONST_MKSA_LAMBERT	cd sr / m^2
footlambert	GSL_CONST_MKSA_FOOTLAMBERT	cd sr / m^2
curie	GSL_CONST_MKSA_CURIE	1 / s
roentgen	GSL_CONST_MKSA_ROENTGEN	A s / kg
rad	GSL_CONST_MKSA_RAD	m^2 / s^2
solar_mass	GSL_CONST_MKSA_SOLAR_MASS	kg
bohr_radius	GSL_CONST_MKSA_BOHR_RADIUS	m
vacuum_permittivity	GSL_CONST_MKSA_VACUUM_PERMITTIVITY	A^2 s^4 / kg m^3

4.3 pygsl.const.cgsm
Scientific constants in cgsm units

```
from pygsl.const import cgsm
print "a teaspoon contains %g ml"%cgs.teaspoon
```

You can access the following constants:

Name	gsl Name	unit or value
speed_of_light	GSL_CONST_CGSM_SPEED_OF_LIGHT	cm / s
gravitational_constant	GSL_CONST_CGSM_GRAVITATIONAL_CONSTANT	cm^3 / g s^ 2
plancks_constant_h	GSL_CONST_CGSM_PLANCKS_CONSTANT_H	g cm^2 / s
plancks_constant_hbar	GSL_CONST_CGSM_PLANCKS_CONSTANT_HBAR	g cm^2 / s
vacuum_permeability	GSL_CONST_CGSM_VACUUM_PERMEABILITY	cm g / A^2 s^2

Name	gsl Name	unit or value
astronomical_unit	GSL_CONST_CGSM_ASTRONOMICAL_UNIT	cm
light_year	GSL_CONST_CGSM_LIGHT_YEAR	cm
parsec	GSL_CONST_CGSM_PARSEC	cm
grav_accel	GSL_CONST_CGSM_GRAV_ACCEL	cm / sˆ2
electron_volt	GSL_CONST_CGSM_ELECTRON_VOLT	g cmˆ2 / sˆ2
mass_electron	GSL_CONST_CGSM_MASS_ELECTRON	g
mass_muon	GSL_CONST_CGSM_MASS_MUON	g
mass_proton	GSL_CONST_CGSM_MASS_PROTON	g
mass_neutron	GSL_CONST_CGSM_MASS_NEUTRON	g
rydberg	GSL_CONST_CGSM_RYDBERG	g cmˆ2 / sˆ2
boltzmann	GSL_CONST_CGSM_BOLTZMANN	g cmˆ2 / K sˆ2
bohr_magneton	GSL_CONST_CGSM_BOHR_MAGNETON	A cmˆ2
nuclear_magneton	GSL_CONST_CGSM_NUCLEAR_MAGNETON	A cmˆ2
electron_magnetic_moment	GSL_CONST_CGSM_ELECTRON_MAGNETIC_MOMENT	A cmˆ2
proton_magnetic_moment	GSL_CONST_CGSM_PROTON_MAGNETIC_MOMENT	A cmˆ2
molar_gas	GSL_CONST_CGSM_MOLAR_GAS	g cmˆ2 / K mol sˆ2
standard_gas_volume	GSL_CONST_CGSM_STANDARD_GAS_VOLUME	cmˆ3 / mol
minute	GSL_CONST_CGSM_MINUTE	s
hour	GSL_CONST_CGSM_HOUR	s
day	GSL_CONST_CGSM_DAY	s
week	GSL_CONST_CGSM_WEEK	s
inch	GSL_CONST_CGSM_INCH	cm
foot	GSL_CONST_CGSM_FOOT	cm
yard	GSL_CONST_CGSM_YARD	cm
mile	GSL_CONST_CGSM_MILE	cm
nautical_mile	GSL_CONST_CGSM_NAUTICAL_MILE	cm
fathom	GSL_CONST_CGSM_FATHOM	cm
mil	GSL_CONST_CGSM_MIL	cm
point	GSL_CONST_CGSM_POINT	cm
texpoint	GSL_CONST_CGSM_TEXPOINT	cm
micron	GSL_CONST_CGSM_MICRON	cm
angstrom	GSL_CONST_CGSM_ANGSTROM	cm
hectare	GSL_CONST_CGSM_HECTARE	cmˆ2
acre	GSL_CONST_CGSM_ACRE	cmˆ2
barn	GSL_CONST_CGSM_BARN	cmˆ2
liter	GSL_CONST_CGSM_LITER	cmˆ3
us_gallon	GSL_CONST_CGSM_US_GALLON	cmˆ3
quart	GSL_CONST_CGSM_QUART	cmˆ3
pint	GSL_CONST_CGSM_PINT	cmˆ3
cup	GSL_CONST_CGSM_CUP	cmˆ3
fluid_ounce	GSL_CONST_CGSM_FLUID_OUNCE	cmˆ3
tablespoon	GSL_CONST_CGSM_TABLESPOON	cmˆ3
teaspoon	GSL_CONST_CGSM_TEASPOON	cmˆ3
canadian_gallon	GSL_CONST_CGSM_CANADIAN_GALLON	cmˆ3
uk_gallon	GSL_CONST_CGSM_UK_GALLON	cmˆ3
miles_per_hour	GSL_CONST_CGSM_MILES_PER_HOUR	cm / s
kilometers_per_hour	GSL_CONST_CGSM_KILOMETERS_PER_HOUR	cm / s
knot	GSL_CONST_CGSM_KNOT	cm / s
pound_mass	GSL_CONST_CGSM_POUND_MASS	g
ounce_mass	GSL_CONST_CGSM_OUNCE_MASS	g
ton	GSL_CONST_CGSM_TON	g
metric_ton	GSL_CONST_CGSM_METRIC_TON	g
uk_ton	GSL_CONST_CGSM_UK_TON	g

Name	gsl Name	unit or value
troy_ounce	GSL_CONST_CGSM_TROY_OUNCE	g
carat	GSL_CONST_CGSM_CARAT	g
unified_atomic_mass	GSL_CONST_CGSM_UNIFIED_ATOMIC_MASS	g
gram_force	GSL_CONST_CGSM_GRAM_FORCE	cm g / s^2
pound_force	GSL_CONST_CGSM_POUND_FORCE	cm g / s^2
kilopound_force	GSL_CONST_CGSM_KILOPOUND_FORCE	cm g / s^2
poundal	GSL_CONST_CGSM_POUNDAL	cm g / s^2
calorie	GSL_CONST_CGSM_CALORIE	g cm^2 / s^2
btu	GSL_CONST_CGSM_BTU	g cm^2 / s^2
therm	GSL_CONST_CGSM_THERM	g cm^2 / s^2
horsepower	GSL_CONST_CGSM_HORSEPOWER	g cm^2 / s^3
bar	GSL_CONST_CGSM_BAR	g / cm s^2
std_atmosphere	GSL_CONST_CGSM_STD_ATMOSPHERE	g / cm s^2
torr	GSL_CONST_CGSM_TORR	g / cm s^2
meter_of_mercury	GSL_CONST_CGSM_METER_OF_MERCURY	g / cm s^2
inch_of_mercury	GSL_CONST_CGSM_INCH_OF_MERCURY	g / cm s^2
inch_of_water	GSL_CONST_CGSM_INCH_OF_WATER	g / cm s^2
psi	GSL_CONST_CGSM_PSI	g / cm s^2
poise	GSL_CONST_CGSM_POISE	g / cm s
stokes	GSL_CONST_CGSM_STOKES	cm^2 / s
faraday	GSL_CONST_CGSM_FARADAY	A s / mol
electron_charge	GSL_CONST_CGSM_ELECTRON_CHARGE	A s
gauss	GSL_CONST_CGSM_GAUSS	g / A s^2
stilb	GSL_CONST_CGSM_STILB	cd / cm^2
lumen	GSL_CONST_CGSM_LUMEN	cd sr
lux	GSL_CONST_CGSM_LUX	cd sr / cm^2
phot	GSL_CONST_CGSM_PHOT	cd sr / cm^2
footcandle	GSL_CONST_CGSM_FOOTCANDLE	cd sr / cm^2
lambert	GSL_CONST_CGSM_LAMBERT	cd sr / cm^2
footlambert	GSL_CONST_CGSM_FOOTLAMBERT	cd sr / cm^2
curie	GSL_CONST_CGSM_CURIE	1 / s
roentgen	GSL_CONST_CGSM_ROENTGEN	A s / g
rad	GSL_CONST_CGSM_RAD	cm^2 / s^2
solar_mass	GSL_CONST_CGSM_SOLAR_MASS	g
bohr_radius	GSL_CONST_CGSM_BOHR_RADIUS	cm
vacuum_permittivity	GSL_CONST_CGSM_VACUUM_PERMITTIVITY	A^2 s^4 / g cm^3

4.4 pygsl.const.num
Scientific number constants

```
from pygsl.const import *
# an alternative to
# from pygsl.const.num import *
print "fine structure is 1/137 with an error of %g%%"%(abs(1.0/137.0/fine_structure-1.0)*100.0)
```

Only two constants are available:

Name	gsl Name	unit
fine_structure	GSL_CONST_NUM_FINE_STRUCTURE	1
avogadro	GSL_CONST_NUM_AVOGADRO	1 / mol

CHAPTER

FIVE

pygsl.poly
Polynomials

Wrapper over the functions as described in Chaper 6 of the GNU Scientific Libraryreference manual.

There are routines for finding real and complex roots of quadratic and cubic equations using analytic methods. An iterative polynomial solver is also available for finding the roots of general polynomials with real coefficients (of any order).

eval (*c, a*)
evaluate the equation $c[0] + c[1]x + c[2]x^2 + \cdots + c[len-1]x^{len-1}$ using Horner's method for stability at *x*. "len" is the number of coefficients (equal to the length of the python object)

class dd (*xa,ya*)
This class computes a divided-difference representation of the interpolating polynomial for the points (*xa*, *ya*).

get_dd ()
Get the devided-difference representation

eval (*x*)
evaluate the representation at *x*

taylor (*x*)
convert the internal representation to a Taylor expansion

solve_quadratic (*a,b,c*)
computes the real roots of the equation $a \cdot x^2 + b \cdot x + c = 0$ all variables are real varibales using a "C double type" for internal representation. returns the number of roots and their value.

complex_solve_quadratic (*a,b,c*)
computes the complex roots of the equation $a \cdot x^2 + b \cdot x + c = 0$. all variables are complex varibales. Returns the number of roots and their value.

solve_cubic (*a,b,c*)
computes the real roots of the equation $x^3 + a \cdot x^2 + b \cdot x + c = 0$. All variables are real varibales using a "C double type" for internal representation. Returns the number of roots and their value.

complex_solve_cubic (*a,b,c*)
computes the complex roots of the equation $x^3 + a \cdot x^2 + b \cdot x + c = 0$. all variables are complex variables. returns the number of roots and their value.

class poly_complex (*n*)
intialise the class giving the dimension of the problem

solve (*a*)
This method computes the roots of the general polynomial $P(x) = a0 + a1x + a2x^2 + ... + an - 1x^{n-1}$ using balanced-QR reduction of the companion matrix. The parameter n specifies the length of the coefficient array. The coefficient of the highest order term must be non-zero. The n-1 roots are returned in a

complex array.
The method raises GSL_EFAILED if the QR reduction does not converge.

CHAPTER

SIX

pygsl.chebyshev

class cheb_series()
This base class can be instantiated by its name

```
import pygsl.chebyshev
s=pygsl.chebyshev.cheb_series()
```

__init__(*n*)
n ... number of coefficients

init(*f, a, b*)
This function computes the Chebyshev approximation for the function F over the range (a,b) to the previously specified order. The computation of the Chebyshev approximation is an O(n^2) process, and requires n function evaluations.

f ... a gsl_function a ... lower limit b ... upper limit

eval(*x*)
This function evaluates the Chebyshev series at a given point X.

eval_err(*x*)
This function computes the Chebyshev series at a given point X, estimating both the series RESULT and its absolute error ABSERR. The error estimate is made from the first neglected term in the series.

eval_n(*n, x*)
This function evaluates the Chebyshev series at a given point x, to (at most) the given order n

eval_n_err(*n, x*)
This function evaluates a Chebyshev series at a given point X, estimating both the series RESULT and its absolute error ABSERR, to (at most) the given order ORDER. The error estimate is made from the first neglected term in the series.

calc_deriv()
This method computes the derivative of the series CS. It returns a new instance of the cheb_series class.

calc_integ()
This method computes the integral of the series CS. It returns a new instance of the cheb_series class.

get_a()
Get the lower boundary of the current representation

get_b()
Get the upper boundary of the current representation

get_coefficients()
Get the chebyshev coefficients.

get_f()
Get the value f (what is it ?) The documentation does not tell anything about it.

get_order_sp()
Get the value f (what is it ?) The documentation does not tell anything about it.

set_a()
Set the lower boundary of the current representation

set_b()
Set the upper boundary of the current

set_coefficients()
Sets the chebyshev coefficients.

set_f(*f*)
Set the value f (what is it ?)

set_order_sp(...)
Set the value f (what is it ?)

gsl_function(*f, params*)
This class defines the callbacks known as gsl_function to gsl.

e.g to supply the function f:

def f(x, params): a = params[0] b = parmas[1] c = params[3] return a * x ** 2 + b * x + c

to some solver, use

function = gsl_function(f, params)

CHAPTER

SEVEN

pygsl.deriv
Numerical Differentiation

This chapter describes the available functions for numerical differentiation.

The functions described in this chapter compute numerical derivatives by finite differencing. An adaptive algorithm is used to find the best choice of finite difference and to estimate the error in the derivative. This module supersedes the diff module which has been deprecated with the release of GSL-1. XXX

central (*func, x, h*)
This function computes the numerical derivative of the function *func* at the point *x* using an adaptive central difference algorithm with a step size of h. A tuple `(result, error)` is returned with the derivative and its estimated absolute error.

backward (*func, x, h*)
This function computes the numerical derivative of the function *func* at the point *x* using an adaptive backward difference algorithm with a step size of h. The function *func* is evaluated only at points smaller than *x* and at *x* itself. A tuple `(result, error)` is returned with the derivative and its estimated absolute error.

forward (*func, x, h*)
This function computes the numerical derivative of the function *func* at the point *x* using an adaptive forward difference algorithm with a step size of h. The function *func* is evaluated only at points greater than *x* and at *x* itself. A tuple `(result, error)` is returned with the derivative and its estimated absolute error.

See Also:

The algorithms used by these functions are described in the following book:

S.D. Conte and Carl de Boor, *Elementary Numerical Analysis: An Algorithmic Approach*, McGraw-Hill, 1972.

CHAPTER

EIGHT

pygsl.histogram
Histogram Types

This chapter is about the `histogram` and `histogram2d` type that are contained in `pygsl.histogram`.

8.1 `histogram` — 1-dimensional histograms

class histogram(*long size | histogram h*)
This type implements all methods on `struct gsl_histogram`.

alloc(*long length*)
allocate necessary space, returns `None`

set_ranges_uniform(*float upper, float lower*)
set the ranges to uniform distance, returns `None`

reset()
sets all bin values to 0, returns `None`

increment(*float x*)
increments corresponding bin, returns `None`

accumulate(*float x, float weight*)
adds the weight to corresponding bin, returns `None`

max()
returns upper range, as `float`

min()
returns lower range, as `float`

bins()
returns number of bins, as `long`

get(*long n*)
gets value of indexed bin, returns `float`

get_range(*long n*)
gets upper and lower range of indexed bin, returns `(float,float)`

find(*float x*)
finds index of corresponding bin, returns `long`

max_val()
returns maximal bin value, as `float`

max_bin()
returns bin index with maximal value, as `long`

min_val()
returns minimal bin value, as `float`

min_bin()
returns bin index with minimal value, as `long`

mean()
returns mean of histogram, as `float`

sigma()
returns std deviation of histogram, as `float`

sum()
returns sum of bin values, as `float`

set_ranges(*sequence ranges*)
sets range according given sequence, returns `None`

shift(*float offset*)
shifts the contents of the bins by the given offset, returns `None`

scale(*float scale*)
multiplies the contents of the bins by the given scale, returns `None`

equal_bins_p()
true if the all of the individual bin ranges are identical, returns `int`

add(*histogram h*)
adds the contents of the bins, returns `None`

sub(*histogram h*)
substracts the contents of the bins, returns `None`

mul(*histogram h*)
multiplicates the contents of the bins, returns `None`

div(*histogram h*)
divides the contents of the bins, returns `None`

clone(*histogram h*)
returns a new copy of this histogram, returns new `histogram`

copy(*histogram h*)
copies the given histogram to myself, returns `None`

read(*file input*)
reads histogram binary data from file, returns `None`

write(*file output*)
writes histogram binary data to file, returns `None`

scanf(*file input*)
reads histogram data from file using scanf, returns `None`

printf(*file output*)
writes histogram data to file using printf, returns `None`

Some mapping operations are supported, too:

Mapping syntax	Effect
`histogram[index]`	returns the value of the indexed bin
`histogram[index]=value`	sets the value of the indexed bin
`len(histogram)`	returns the length of the histogram

See Also:

For the special meaning and details please consult the GNU Scientific Library reference.

8.2 `histogram2d` — 2-dimensional histograms

class histogram2d(*long size x, long size y | histogram2d h*)
This class holds a 2d array and 2 sets of ranges for x and y coordinates for a two paramter statistical event. It can be constructed by size parameters or as a copy from another histogram. Most of the methods are the same as of `histogram`.

set_ranges_uniform(*float xmin, float xmax, float ymin, float ymax*)
set the ranges to uniform distance, returns `None`

alloc(*long nx, long ny*)
allocate necessary space, returns `None`

reset()
sets all bin values to 0, returns `None`

increment(*float x, float y*)
increments corresponding bin, returns `None`

accumulate(*float x, float y, float weight*)
adds the weight to corresponding bin, returns `None`

xmax()
returns upper x range as `float`

xmin()
returns lower x range as `float`

ymax()
returns upper y range as `float`

ymin()
returns lower y range as `float`

nx()
returns number of x bins as `long`

ny()
returns number of y bins as `long`

get(*long i, long j*)
gets value of indexed bin, returns `float`

get_xrange(*long i*)
gets upper and lower x range of indexed bin, returns `(float lower, float upper)`

get_yrange(*long j*)
gets upper and lower y range of indexed bin, returns `(float lower, float upper)`

find(*float x, float y*)
finds index pair of corresponding value pair, returns (`long`,`long`)

max_val()
returns maximal bin value as `float`

max_bin()
returns bin index with maximal value as `long`

min_val()
returns minimal bin value as `float`

min_bin()
returns bin index with minimal value as `long`

sum()
returns sum of bin values as `float`

xmean()
returns x mean of histogram as `float`

xsigma()
returns x std deviation of histogram as `float`

ymean()
returns y mean of histogram as`float`

ysigma()
returns y std deviation of histogram as `float`

cov()
returns covariance of histogram as `float`

set_ranges(*sequence xranges, sequence yranges*)
set the ranges according to given sequences, returns `None`

shift(`float` *offset*)
shifts the contents of the bins by the given offset, returns `None`

scale(`float` *scale*)
multiplies the contents of the bins by the given scale, returns `None`

equal_bins_p()
true if the all of the individual bin ranges are identical, returns `int`

add(`histogram2d` *h*)
adds the contents of the bins, returns `None`

sub(`histogram2d` *h*)
substracts the contents of the bins, returns `None`

mul(`histogram2d` *h*)
multiplicates the contents of the bins, returns `None`

div(`histogram2d` *h*)
divides the contents of the bins, returns `None`

clone()
returns a copy instance of `histogram2d`

copy(`histogram2d` *h*)
copies the given histogram to myself, returns `None`

read(*file input*)
reads histogram binary data from file, returns `None`

writew(*file output*)
writes histogram binary data to file, returns `None`

scanf (*file input*)
reads histogram data from file using scanf, returns `None`

printf (*file input*)
writes histogram data to file using printf, returns `None`

Some mapping operations are supported, too:

Mapping syntax	Effect
`histogram[x_index,y_index]`	returns the value of the indexed bin
`histogram[x_index,y_index]=value`	sets the value of the indexed bin
`len(histogram)`	returns the size of the histogram, i.e nx×ny

See Also:

For the special meaning and details please consult the GNU Scientific Library reference.

8.3 `histogram_pdf` and `histogram2d_pdf`

To be implemented...

CHAPTER

NINE

pygsl.rng
Random Number Generators

This chapter introduces the random number generator type provided by `pygsl`.

9.1 Random Number Generators

All random number generatores are the same python type (PyGSL_rng), but using the approbriate GSL random generator for generating the random numbers. Use the method `name` to get the name of the rng used internally.

Methods of this type rng provide the transformation to different probability distributions and give access to basic properties of random number generators. All methods allow to pass one optional integer. Then the method will be evaluated n times and the result will be returned as an array.

pytype rng (*string typenamme | rng r*)

This base class can be instantiated by its name

```
import pygsl.rng
my_ran0=pygsl.rng.ran0()
```

.

The type of the allocated generator is given by the method

name ()

which returns its name as string.

All generators can be seeded with

set (*seed*)

which sets the internal seed according to the positive integer `seed`. Zero as seed has a special meaning, please read details in the gsl reference.

The basic returned number type is integer, these are generated by

get ()

which returns the next number of the pseudo random sequence.

All methods support internal sampling; i.e each method has an optional integer. If given it will return a sample of the approbriate size.

get (*—n*)

will return the next n numbers of the pseudo random sequence.

Basic information about these numbers can be obtained by

max ()

maximum number of this sequence and

min ()

minimum number of this sequence.

Implemented uniform probability densities are:

uniform()
: returns a real number between $[0, 1)$.

uniform_pos()
: returns a real number between $(0, 1)$ — this excludes 0.

uniform_int(*upper limit*)
: returns an integer from 0 to the upper limit (exclusive). If this limit is larger than the number of return values of the underlying generator, `pygsl.gsl_Error` is raised.

Furthermore a lot of derived probability densities can be used:

gaussian(*sigma*)
: gaussian distribution with mean 0 and given sigma returns `float`

gaussian_ratio_method(*sigma*)
: gaussian distribution with mean 0 and given sigma. This variate uses the Kinderman-Monahan ratio method. returns `float`

ugaussian()
: gaussian distribution with unit sigma and mean 0. returns `float`

ugaussian_ratio_method()
: gaussian distribution with unit sigma and mean 0. This variate uses the Kinderman-Monahan ratio method. returns `float`

gaussian_tail(*sigma, a*)
: upper tail of a Gaussian distribution with standard deviation sigma¿0. returns `float`

ugaussian_tail(*a*)
: upper tail of a Gaussian distribution with unit standard deviation. returns `float`

bivariate_gaussian(*sigma_x, sigma_y, rho*)
: pair of correlated gaussian variates, with mean zero, correlation coefficient rho and standard deviations sigma_x and sigma_y in the x and y directions returns `(float,float)`

exponential(*mu*)
: returns `float`

laplace(*mu*)
: returns `float`

exppow(*mu, a*)
: returns `float`

cauchy(*mu*)
: returns `float`

rayleigh(*sigma*)
: returns `float`

rayleigh_tail(*a, sigma*)
: returns `float`

levy(*mu,a*)
: returns `float`

levy_skew(*mu,a,beta*)
: returns `float`

gamma(*a, b*)
: returns `float`

gamma_int(*long a*)

returns `float`

flat(*a, b*)
returns `float`

lognormal(*zeta, sigma*)
returns `float`

chisq(*nu*)
returns `float`

fdist(*nu1, nu2*)
returns `float`

tdist(*nu*)
returns `float`

beta(*a, b*)
returns `float`

logistic(*mu*)
returns `float`

pareto(*a, b*)
returns `float`

dir_2d()
returns `(float, float)`

dir_2d_trig_method()
returns `(float, float)`

dir_3d()
returns `(float, float, float)`

dir_nd(*int n*)
returns `(float, ..., float)`

weibull(*mu, a*)
returns `float`

gumbel1(*a, b*)
returns `float`

gumbel2()

poisson()

bernoulli()

binomial()

negative_binomial()

pascal()

geometric()

hypergeometric()

logarithmic()

landau()

erlang()

The different generator classes are created according to the output of `gsl_rng_types_setup()` when the `pygsl.rng` is loaded. Here is the list of children from `rng` for gsl-1.2:

rng_borosh13, rng_coveyou, rng_cmrg, rng_fishman18, rng_fishman20, rng_fishman2x, rng_gfsr4, rng_knuthran, rng_knuthran2, rng_lecuyer21, rng_minstd, rng_mrg, rng_mt19937, rng_mt19937_1999, rng_mt19937_1998, rng_r250, rng_ran0, rng_ran1, rng_ran2, rng_ran3, rng_rand, rng_rand48, rng_random128_bsd, rng_random128_glibc2, rng_random128_libc5, rng_random256_bsd, rng_random256_glibc2, rng_random256_libc5, rng_random32_bsd, rng_random32_glibc2, rng_random32_libc5, rng_random64_bsd, rng_random64_glibc2, rng_random64_libc5, rng_random8_bsd, rng_random8_glibc2, rng_random8_libc5, rng_random_bsd, rng_random_glibc2, rng_random_libc5, rng_randu, rng_ranf, rng_ranlux, rng_ranlux389, rng_ranlxd1, rng_ranlxd2, rng_ranlxs0, rng_ranlxs1, rng_ranlxs2, rng_ranmar, rng_slatec, rng_taus, rng_taus2, rng_taus113, rng_transputer, rng_tt800, rng_uni, rng_uni32, rng_vax, rng_waterman14, and rng_zuf.

The default generator of the rng defaults to rng_mt19937 but can be set from the environment variable GSL_RNG_TYPE using the function rng.env_setup().

9.2 Probability Density Functions

9.3 Using probability densities with random number generators

CHAPTER

TEN

pygsl.sum
Series acceleration

This chapter describes the use of the series acceleration tools based on the Levin u-transform. This method takes a small number of terms from the start of a series and uses a systematic approximation to compute an extrapolated value and an estimate of its error. The u-transform works for both convergent and divergent series, including asymptotic series.

$$\texttt{levin\_sum(a)} = (A, \epsilon) \qquad \text{where} \qquad A \approx \sum n = 0^{\infty} an \pm \epsilon, \tag{10.1}$$

$\texttt{a} = [a0, a1, \ldots, an]$, and ϵ is an estimate of the absolute error.

Note: This function is intended for summing analytic series where each term is known to high accuracy, and the rounding errors are assumed to originate from finite precision. They are taken to be relative errors of order `GSL_DBL_EPSILON` for each term (as defined in the GNU Scientific Library source code).

10.1 Function list

levin_sum(*a, truncate=False, info_dict=None*)

Return (A, ϵ) where A is the approximated sum of the series (10.1) and ϵ is its absolute error estimate.

The calculation of the error in the extrapolated value is an $O(N^2)$ process, which is expensive in time and memory. A full table of intermediate values and derivatives through to $O(N)$ must be computed and stored, but this does give a reliable error estimate.

A faster but less reliable method which estimates the error from the convergence of the extrapolated value is employed if *truncate* is `True`. This attempts to estimate the error from the "truncation error" in the extrapolation, the difference between the final two approximations. Using this method avoids the need to compute an intermediate table of derivatives because the error is estimated from the behavior of the extrapolated value itself. Consequently this algorithm is an $O(N)$ process and only requires $O(N)$ terms of storage. If the series converges sufficiently fast then this procedure can be acceptable. It is appropriate to use this method when there is a need to compute many extrapolations of series with similar convergence properties at high-speed. For example, when numerically integrating a function defined by a parameterized series where the parameter varies only slightly. A reliable error estimate should be computed first using the full algorithm described above in order to verify the consistency of the results.

If a dictionary is passed as *info_dict*, then two entries will be added: *info_dict*`['terms_used']` will be the number of terms used[1] and *info_dict*`['sum_plain']` will be the sum of these terms without acceleration.

[1] Note that it appears that this is the number of terms *beyond* the first term that are used. I.e. there are a total of *terms_used* + 1 terms:

$$\textit{sum\_plain} = \sum n = 0^{\textit{terms\_used}} an \tag{10.2}$$

10.2 Further Reading

For details on the underlying implementation of these functions please consult the GNU Scientific Library reference manual. The algorithms used by these functions are described Fessler *et al.* (1983). The theory of the u-transform was presented Levin in 1973, and a review paper from 2000 by Homeier is available online.

See Also:

T. Fessler, W. F. Ford, D. A. Smith, *hurry: An acceleration algorithm for scalar sequences and series*. ACM Transactions on Mathematical Software, **9**(3):346–354, (1983), and Algorithm 602 9(3):355–357, 1983.

D. Levin, *Development of Non-Linear Transformations for Improving Convergence of Sequences,* Intern. J. Computer Math. **B3**:371–388, (1973).

Herbert H. H. Homeier, Scalar Levin-Type Sequence Transformations.
(`http://arXiv.org/abs/math/0005209`)

CHAPTER

ELEVEN

pygsl.statistics
Statistics functions

This chapter describes the statistical functions in the library. The basic statistical functions include routines to compute the mean, variance and standard deviation. More advanced functions allow you to calculate absolute deviations, skewness, and kurtosis as well as the median and arbitrary percentiles.

The algorithms provided here use recurrence relations to compute average quantities in a stable way, without large intermediate values that might overflow. All functions work on any Python sequence (of appropriate data-type), but see section **??** for advantages and drawbacks of different kinds of input data.

See Also:

For details on the underlying implementation of these functions please consult the GNU Scientific Library reference manual.

11.1 Organization of the module

Individual parts of the GSL functions names, providing artificial namespaces in C, are mapped to modules and submodules in PyGSL. That is, `gsl_stats_mean` can be found as `pygsl.statistics.mean` and `gsl_stats_long_mean` as `pygsl.statistics.long.mean`.

The functions in the module are available in versions for datasets in the standard and NumPy floating-point and integer types. The generic versions available in the `pygsl.statistics` module are using the generic GSL `double` versions. The submodules use GSL functions according to the submodule name, e.g. long for `pygsl.statistics.long`.

Implemented submodules are `char`, `uchar`, `short`, `int`, `long`, `float`, and `double`. The latter one also serves as default and is used whenever you don't expclicitely state a different datatype. In most cases it is appropriate to simply use the default implementation as it covers the widest range of the real space, offers high precision, and as such is simple to use. If you have a sequence of all integer values it is straightforward to use `pygsl.statistics.long` functions as these use an implementation corresponding to Pythons `Float`-type. These implemented submodules represent all numeric datatypes available in Python (`Int`, `Float`) besides `Long Int` which has no representation in standard C, as well as all numeric datatypes available in NumPy that have corresponding implementations in GSL (on 32 bit systems these are: Character, UnsigendInt8, Int16, Int32, Int, Float32, Float).

11.2 Available functions

11.2.1 Mean, Standard Deviation, and Variance

mean (*x*)
Arithmetic mean (*sample mean*) of *x*:

$$\hat{\mu} = \frac{1}{N} \sum xi \tag{11.1}$$

variance (*x*)
Estimated (*sample*) variance of *x*:

$$\hat{\sigma}^2 = \frac{1}{N-1} \sum (xi - \hat{\mu})^2 \tag{11.2}$$

This function computes the mean via a call to `mean`. If you have already computed the mean then you can pass it directly to `variance_m`.

variance_m (*x, mean*)
Estimated (*sample*) variance of *x* relative to *mean*:

$$\hat{\sigma}^2 = \frac{1}{N-1} \sum (xi - mean)^2 \tag{11.3}$$

sd (*x*)

sd_m (*x, mean*)
The standard deviation is defined as the square root of the variance of *x*. These functions returns the square root of the respective variance-functions above.

variance_with_fixed_mean (*x, mean*)
Compute an unbiased estimate of the variance of *x* when the population mean *mean* of the underlying distribution is known *a priori*. In this case the estimator for the variance uses the factor $1/N$ and the sample mean $\hat{\mu}$ is replaced by the known population mean μ:

$$\hat{\sigma}^2 = \frac{1}{N} \sum (xi - \mu)^2 \tag{11.4}$$

11.2.2 Absolute deviation

absdev (*data*)
Compute the absolute deviation from the mean of *data* The absolute deviation from the mean is defined as

$$absdev = (1/N) \sum |xi - \hat{\mu}| \tag{11.5}$$

where xi are the elements of the dataset *data*. The absolute deviation from the mean provides a more robust measure of the width of a distribution than the variance. This function computes the mean of *data* via a call to `mean`.

absdev_m (*data, mean*)
Compute the absolute deviation of the dataset *data* relative to the given value of *mean*

$$absdev = (1/N) \sum |xi - mean| \tag{11.6}$$

This function is useful if you have already computed the mean of *data* (and want to avoid recomputing it), or wish to calculate the absolute deviation relative to another value (such as zero, or the median).

11.2.3 Higher moments (skewness and kurtosis)

skew (*data*)
Compute the skewness of *data*. The skewness is defined as

$$skew = (1/N) \sum ((xi - \hat{\mu})/\hat{\sigma})^3 \tag{11.7}$$

where xi are the elements of the dataset *data*. The skewness measures the asymmetry of the tails of a distribution.

The function computes the mean and estimated standard deviation of *data* via calls to `mean` and `sd`.

skew_m_sd (*data, mean, sd*)
Compute the skewness of the dataset *data* using the given values of the mean *mean* and standard deviation varsd

$$skew = (1/N) \sum ((xi - mean)/sd)^3 \tag{11.8}$$

These functions are useful if you have already computed the mean and standard deviation of *data* and want to avoid recomputing them.

kurtosis (*data*)
Compute the kurtosis of *data*. The kurtosis is defined as

$$kurtosis = ((1/N) \sum ((xi - \hat{\mu})/\hat{\sigma})^4) - 3 \tag{11.9}$$

The kurtosis measures how sharply peaked a distribution is, relative to its width. The kurtosis is normalized to zero for a gaussian distribution.

kurtosis_m_sd (*data, mean, sd*)
This function computes the kurtosis of the dataset *data* using the given values of the mean *mean* and standard deviation *sd*

$$kurtosis = ((1/N) \sum ((xi - mean)/sd)^4) - 3 \tag{11.10}$$

This function is useful if you have already computed the mean and standard deviation of *data* and want to avoid recomputing them.

11.2.4 Autocorrelation

lag1_autocorrelation (*x*)
Computes the lag-1 autocorrelation of the dataset *x*

$$a1 = \frac{\sum^n i = 1(xi - \hat{\mu})(xi - 1 - \hat{\mu})}{\sum^n i = 1(xi - \hat{\mu})(xi - \hat{\mu})} \tag{11.11}$$

lag1_autocorrelation_m (*x, mean*)
Computes the lag-1 autocorrelation of the dataset *x* using the given value of the mean *mean*.

$$a1 = \frac{\sum i = 1^n (xi - mean)(xi - 1 - mean)}{\sum^n i = 1(xi - mean)(xi - mean)} \tag{11.12}$$

11.2.5 Covariance

covariance (*x, y*)
Computes the covariance of the datasets *x* and *y* which must be of same length.

$$c = \frac{1}{n-1} \sum^n i = 1(xi - \hat{x})(yi - \hat{y}) \tag{11.13}$$

lag1_autocorrelation_m(*x, y, mean_x, mean_y*)
Computes the covariance of the datasets *x* and *y* using the given values of the means *mean_x* and *mean_y*. The datasets *x* and *y* must be of equal length.

$$c = \frac{1}{n-1} \sum^{n} i = 1(xi - mean\_x)(yi - mean\_y) \tag{11.14}$$

11.2.6 Maximum and Minimum values

max(*data*)
This function returns the maximum value in *data*. The maximum value is defined as the value of the element xi which satisfies $xi \geq xj$ for all j.

If you want instead to find the element with the largest absolute magnitude you will need to apply 'fabs' or 'abs' to your data before calling this function.

min(*data*)
This function returns the minimum value in *data*. The maximum value is defined as the value of the element xi which satisfies $xi \leq xj$ for all j.

If you want instead to find the element with the smallest absolute magnitude you will need to apply 'fabs' or 'abs' to your data before calling this function.

minmax(*data*)
This function returns both the minimum and maximum values of *data*, determined in a single pass.

max_index(*data*)
This function returns the index of the maximum value in *data*. The maximum value is defined as the value of the element xi which satisfies $xi \geq xj$ for all j. When there are several equal maximum elements then the first one is chosen.

min_index(*data*)
This function returns the index of the minimum value in *data*. The minimum value is defined as the value of the element xi which satisfies $xi \leq xj$ for all j. When there are several equal minimum elements then the first one is chosen.

minmax_index(*data*)
This function returns the indexes of the minimum and maximum values of *data*, determined in a single pass.

11.2.7 Median and Percentiles

The median and percentile functions described in this section operate on sorted data. For convenience we use "quantiles", measured on a scale of 0 to 1, instead of percentiles (which use a scale of 0 to 100).

median_from_sorted_data(*data*)
This function returns the median value of *data*. The elements of the array must be in ascending numerical order. There are no checks to see whether the data are sorted, so the function `sort` should always be used first.

When the dataset has an odd number of elements the median is the value of element (n-1)/2. When the dataset has an even number of elements the median is the mean of the two nearest middle values, elements (n-1)/2 and n/2. Since the algorithm for computing the median involves interpolation this function always returns a floating-point number, even for integer data types.

quantile_from_sorted_data(*data, F*)
This function returns a quantile value of *data*. The elements of the array must be in ascending numerical order. The quantile is determined by the *F*, a fraction between 0 and 1. For example, to compute the value of the 75th percentile *F* should have the value 0.75.

There are no checks to see whether the data are sorted, so the function `sort` should always be used first.

The quantile is found by interpolation, using the formula

$$quantile = (1 - \delta)xi + \delta xi + 1 \tag{11.15}$$

where i is $floor((n-1)f)$ and δ is $(n-1)f - i$.

Thus the minimum value of the array (*data[0]*) is given by *F* equal to zero, the maximum value (*data[-1]*) is given by *F* equal to one and the median value is given by *F* equal to 0.5. Since the algorithm for computing quantiles involves interpolation this function always returns a floating-point number, even for integer data types.

11.2.8 Weighted Samples

The functions described in this section allow the computation of statistics for weighted samples. The functions accept an array of samples, xi, with associated weights, wi. Each sample xi is considered as having been drawn from a Gaussian distribution with variance σi^2. The sample weight wi is defined as the reciprocal of this variance, $wi = 1/\sigma i^2$. Setting a weight to zero corresponds to removing a sample from a dataset.

wmean (*w, data*)
This function returns the weighted mean of the dataset *data* using the set of weights *w*. The weighted mean is defined as

$$\hat{\mu} = (\sum wixi)/(\sum wi) \tag{11.16}$$

wvariance (*w, data*)
This function returns the estimated variance of the dataset *data*, using the set of weights *w*. The estimated variance of a weighted dataset is defined as

$$\hat{\sigma}^2 = ((\sum wi)/((\sum wi)^2 - \sum(wi^2))) \sum wi(xi - \hat{\mu})^2 \tag{11.17}$$

Note that this expression reduces to an unweighted variance with the familiar $1/(N-1)$ factor when there are N equal non-zero weights.

wvariance_m (*w, data, wmean*)
This function returns the estimated variance of the weighted dataset *data* using the given weighted mean *wmean*.

wsd (*w, data*)
The standard deviation is defined as the square root of the variance. This function returns the square root of the corresponding variance function `wvariance` above.

wsd_m (*w, data, wmean*)
This function returns the square root of the corresponding variance function `wvariance_m` above.

wvariance_with_fixed_mean (*w, data, mean*)
This function computes an unbiased estimate of the variance of weighted dataset *data* when the population mean *mean* of the underlying distribution is known _a priori_. In this case the estimator for the variance replaces the sample mean $\hat{\mu}$ by the known population mean μ,

$$\hat{\sigma}^2 = (\sum wi(xi - \mu)^2)/(\sum wi) \tag{11.18}$$

wsd_with_fixed_mean (*w, data, mean*)
The standard deviation is defined as the square root of the variance. This function returns the square root of the corresponding variance function above.

wabsdev (*w, data*)
This function computes the weighted absolute deviation from the weighted mean of *data*. The absolute deviation from the mean is defined as

$$absdev = (\sum wi|xi - \hat{\mu}|)/(\sum wi) \tag{11.19}$$

wabsdev_m (*w, data, wmean*)
This function computes the absolute deviation of the weighted dataset DATA about the given weighted mean WMEAN.

wskew(*w, data*)
This function computes the weighted skewness of the dataset DATA.

$$skew = (\sum wi((xi - xbar)/\sigma)^3)/(\sum wi) \tag{11.20}$$

wskew_m_sd(*w, data, mean, wsd*)
This function computes the weighted skewness of the dataset *data* using the given values of the weighted mean and weighted standard deviation, *wmean* and *wsd*.

wkurtosis(*w, data*)
This function computes the weighted kurtosis of the dataset *data*. The kurtosis is defined as

$$kurtosis = ((\sum wi((xi - xbar)/sigma)^4)/(\sum wi)) - 3 \tag{11.21}$$

wkurtosis_m_sd(*w, data, mean, wsd*)
This function computes the weighted kurtosis of the dataset *data* using the given values of the weighted mean and weighted standard deviation, *wmean* and *wsd*.

11.3 Further Reading

See the GSL reference manual for a description of all available functions and the calculations they perform.

The standard reference for almost any topic in statistics is the multi-volume *Advanced Theory of Statistics* by Kendall and Stuart. Many statistical concepts can be more easily understood by a Bayesian approach. The book by Gelman, Carlin, Stern and Rubin gives a comprehensive coverage of the subject. For physicists the Particle Data Group provides useful reviews of Probability and Statistics in the "Mathematical Tools" section of its Annual Review of Particle Physics.

See Also:

Maurice Kendall, Alan Stuart, and J. Keith Ord: *The Advanced Theory of Statistics* (multiple volumes) reprinted as *Kendall's Advanced Theory of Statistics*. Wiley, ISBN 047023380X.

Andrew Gelman, John B. Carlin, Hal S. Stern, Donald B. Rubin: *Bayesian Data Analysis*. Chapman & Hall, ISBN 0412039915.

R.M. Barnett et al: Review of Particle Properties. *Physical Review* **D54**, 1 (1996).

D.E. Groom et al., *The European Physical Journal* **C15**, 1 (2000) and *2001 off-year partial update for the 2002 edition* available on the PDG WWW pages (URL: `http://pdg.lbl.gov/`).

Siegmund Brandt: *Datenanalyse*, 4th ed. 1999, Spektrum, Heidelberg, ISBN 3827401585.

Siegmund Brandt: *Data Analysis*. 3rd ed. 1998, Springer, Berlin, ISBN 0387984984.

CHAPTER

TWELVE

pygsl.testing
Modules in Testing

Modules in this package are often reimplementations of an original package with significant change to the original. The current rng implementation, for example, started its life here. The sf module implemented here will supersede the sf package in one of the next releases. Concerning the other modules the usage is encouraged for tests to see if they work, but use them with caution in your production code!

12.1 pygsl.testing.sf
Special Functions as UFuncs

This chapter provides mainly NumPy UFuncs over the special functions. This means that all input variable can be arrays, and the UFunc will evaluate the gsl function for all its inputs. It is meant to replace the sf module later; please use it and find out if it is useful for you. Only the python specific part is described here. For a general description of the function please see the GSL Reference document.

12.2 UFuncs

These UFuncs allow to evaluate an array of doubles or an array of floats typically. Note that the return values of the functions ending with an _e are not turned to exceptions. Instead the error codes are returned as an separate array. So if you are interested in them you have to do check the error code yourself.

Chi(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

Chi_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

Ci(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

Ci_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

Shi(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

Shi_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

Si(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

Si_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

airy_Ai(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

Argument 2 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX

airy_Ai_deriv(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

Argument 2 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX

airy_Ai_deriv_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

Argument 2 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

airy_Ai_deriv_scaled(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

Argument 2 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX

airy_Ai_deriv_scaled_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

Argument 2 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

airy_Ai_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

Argument 2 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

airy_Ai_scaled(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

Argument 2 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX

airy_Ai_scaled_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

Argument 2 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

airy_Bi(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

Argument 2 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX

airy_Bi_deriv(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

Argument 2 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX

airy_Bi_deriv_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

Argument 2 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

airy_Bi_deriv_scaled(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

Argument 2 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX

airy_Bi_deriv_scaled_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

Argument 2 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

airy_Bi_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

Argument 2 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

airy_Bi_scaled(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

Argument 2 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX

airy_Bi_scaled_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

Argument 2 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

airy_zero_Ai(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

airy_zero_Ai_deriv(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

airy_zero_Ai_deriv_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

airy_zero_Ai_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

airy_zero_Bi(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

airy_zero_Bi_deriv(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

airy_zero_Bi_deriv_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

airy_zero_Bi_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

angle_restrict_pos(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

angle_restrict_pos_err_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

angle_restrict_symm(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

angle_restrict_symm_err_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

atanint(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

atanint_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_I0(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

bessel_I0_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_I0_scaled(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

bessel_I0_scaled_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_I1(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

bessel_I1_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_I1_scaled(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

bessel_I1_scaled_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_In(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

bessel_In_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

bessel_In_scaled(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

bessel_In_scaled_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

bessel_Inu(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

bessel_Inu_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

bessel_Inu_scaled(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

bessel_Inu_scaled_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

bessel_J0(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

bessel_J0_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_J1(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

bessel_J1_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_Jn(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

bessel_Jn_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

bessel_Jnu(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

bessel_Jnu_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

bessel_K0(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

bessel_K0_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_K0_scaled(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

bessel_K0_scaled_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_K1(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

bessel_K1_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_K1_scaled(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

bessel_K1_scaled_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_Kn(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

bessel_Kn_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

bessel_Kn_scaled(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

bessel_Kn_scaled_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

bessel_Knu(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

bessel_Knu_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

bessel_Knu_scaled(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

bessel_Knu_scaled_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

bessel_Y0(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

bessel_Y0_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_Y1(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

bessel_Y1_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_Yn(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

bessel_Yn_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

bessel_Ynu(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

bessel_Ynu_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

bessel_i0_scaled(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

bessel_i0_scaled_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_i1_scaled(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

bessel_i1_scaled_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_i2_scaled(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

bessel_i2_scaled_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_il_scaled(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

bessel_il_scaled_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

bessel_j0(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

bessel_j0_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_j1(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

bessel_j1_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_j2(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

bessel_j2_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_jl (...)
Number of Input Arguments: 2 Number of Output Arguments: 1

bessel_jl_e (...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

bessel_k0_scaled (...)
Number of Input Arguments: 1 Number of Output Arguments: 1

bessel_k0_scaled_e (...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_k1_scaled (...)
Number of Input Arguments: 1 Number of Output Arguments: 1

bessel_k1_scaled_e (...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_k2_scaled (...)
Number of Input Arguments: 1 Number of Output Arguments: 1

bessel_k2_scaled_e (...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_kl_scaled (...)
Number of Input Arguments: 2 Number of Output Arguments: 1

bessel_kl_scaled_e (...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

bessel_lnKnu (...)
Number of Input Arguments: 2 Number of Output Arguments: 1

bessel_lnKnu_e (...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

bessel_y0 (...)
Number of Input Arguments: 1 Number of Output Arguments: 1

bessel_y0_e (...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_y1(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

bessel_y1_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_y2(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

bessel_y2_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_yl(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

bessel_yl_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

bessel_zero_J0(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

bessel_zero_J0_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_zero_J1(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

bessel_zero_J1_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

bessel_zero_Jnu(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

bessel_zero_Jnu_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

beta(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

beta_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

beta_inc(...)
Number of Input Arguments: 3 Number of Output Arguments: 1

beta_inc_e(...)
Number of Input Arguments: 3 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 3 of the C argument list

choose(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

choose_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

clausen(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

clausen_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

conicalP_0(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

conicalP_0_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

conicalP_1(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

conicalP_1_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

conicalP_cyl_reg(...)
Number of Input Arguments: 3 Number of Output Arguments: 1

conicalP_cyl_reg_e(...)
Number of Input Arguments: 3 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 3 of the C argument list

conicalP_half(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

conicalP_half_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

conicalP_mhalf(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

conicalP_mhalf_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

conicalP_sph_reg(...)
Number of Input Arguments: 3 Number of Output Arguments: 1

conicalP_sph_reg_e(...)
Number of Input Arguments: 3 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 3 of the C argument list

cos(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

cos_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

cos_err_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

coulomb_CL_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

coulomb_wave_FG_e(...)
Number of Input Arguments: 4 Number of Output Arguments: 10

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 4 of the C argument list Return Arguments 3 and 4 resemble a gsl_result argument, which is argument 5 of the C argument list Return Arguments 5 and 6 resemble a gsl_result argument, which is argument 6 of the C argument list Return Arguments 7 and 8 resemble a gsl_result argument, which is argument 7 of the C argument list

coupling_3j(...)
Number of Input Arguments: 6 Number of Output Arguments: 1

coupling_3j_e(...)
Number of Input Arguments: 6 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 6 of the C argument list

coupling_6j(...)
Number of Input Arguments: 6 Number of Output Arguments: 1

coupling_6j_e(...)
Number of Input Arguments: 6 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 6 of the C argument list

coupling_9j(...)
Number of Input Arguments: 9 Number of Output Arguments: 1

coupling_9j_e(...)
Number of Input Arguments: 9 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 9 of the C argument list

coupling_RacahW(...)
Number of Input Arguments: 6 Number of Output Arguments: 1

coupling_RacahW_e(...)
Number of Input Arguments: 6 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 6 of the C argument list

dawson(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

dawson_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

debye_1(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

debye_1_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

debye_2(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

debye_2_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

debye_3(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

debye_3_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

debye_4(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

debye_4_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

dilog(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

dilog_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

doublefact(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

doublefact_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

ellint_D(...)
Number of Input Arguments: 4 Number of Output Arguments: 1

Argument 4 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX

ellint_D_e(...)
Number of Input Arguments: 4 Number of Output Arguments: 2

Argument 4 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 4 of the C argument list

ellint_E(...)
Number of Input Arguments: 3 Number of Output Arguments: 1

Argument 3 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX

ellint_E_e(...)
Number of Input Arguments: 3 Number of Output Arguments: 2

Argument 3 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 3 of the C argument list

ellint_Ecomp(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

Argument 2 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX

ellint_Ecomp_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

Argument 2 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

ellint_F(...)
Number of Input Arguments: 3 Number of Output Arguments: 1

Argument 3 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX

ellint_F_e(...)
Number of Input Arguments: 3 Number of Output Arguments: 2

Argument 3 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 3 of the C argument list

ellint_Kcomp(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

Argument 2 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX

ellint_Kcomp_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

Argument 2 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

ellint_P (...)
Number of Input Arguments: 4 Number of Output Arguments: 1

Argument 4 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX

ellint_P_e (...)
Number of Input Arguments: 4 Number of Output Arguments: 2

Argument 4 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 4 of the C argument list

ellint_RC (...)
Number of Input Arguments: 3 Number of Output Arguments: 1

Argument 3 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX

ellint_RC_e (...)
Number of Input Arguments: 3 Number of Output Arguments: 2

Argument 3 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 3 of the C argument list

ellint_RD (...)
Number of Input Arguments: 4 Number of Output Arguments: 1

Argument 4 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX

ellint_RD_e (...)
Number of Input Arguments: 4 Number of Output Arguments: 2

Argument 4 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 4 of the C argument list

ellint_RF (...)
Number of Input Arguments: 4 Number of Output Arguments: 1

Argument 4 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX

ellint_RF_e (...)
Number of Input Arguments: 4 Number of Output Arguments: 2

Argument 4 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 4 of the C argument list

ellint_RJ (...)
Number of Input Arguments: 5 Number of Output Arguments: 1

Argument 5 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX

ellint_RJ_e (...)
Number of Input Arguments: 5 Number of Output Arguments: 2

Argument 5 is a gsl_mode_t, valid parameters are: sf.PREC_DOUBLE or sf.PREC_SINGLE or sf.PREC_APPROX The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 5 of the C argument list

elljac_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 3

The error flag is discarded.

erf(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

erf_Q(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

erf_Q_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

erf_Z(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

erf_Z_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

erf_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

erfc(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

erfc_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

eta(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

eta_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

eta_int(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

eta_int_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

expint_3(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

expint_3_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

expint_E1(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

expint_E1_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

expint_E1_scaled(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

expint_E1_scaled_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

expint_E2(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

expint_E2_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

expint_E2_scaled(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

expint_E2_scaled_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

expint_Ei(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

expint_Ei_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

expint_Ei_scaled(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

expint_Ei_scaled_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

fact(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

fact_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

fermi_dirac_0 (...)
Number of Input Arguments: 1 Number of Output Arguments: 1

fermi_dirac_0_e (...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

fermi_dirac_1 (...)
Number of Input Arguments: 1 Number of Output Arguments: 1

fermi_dirac_1_e (...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

fermi_dirac_2 (...)
Number of Input Arguments: 1 Number of Output Arguments: 1

fermi_dirac_2_e (...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

fermi_dirac_3half (...)
Number of Input Arguments: 1 Number of Output Arguments: 1

fermi_dirac_3half_e (...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

fermi_dirac_half (...)
Number of Input Arguments: 1 Number of Output Arguments: 1

fermi_dirac_half_e (...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

fermi_dirac_inc_0 (...)
Number of Input Arguments: 2 Number of Output Arguments: 1

fermi_dirac_inc_0_e (...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

fermi_dirac_int (...)
Number of Input Arguments: 2 Number of Output Arguments: 1

fermi_dirac_int_e (...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

fermi_dirac_m1(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

fermi_dirac_m1_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

fermi_dirac_mhalf(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

fermi_dirac_mhalf_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

gamma(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

gamma_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

gamma_inc_P(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

gamma_inc_P_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

gamma_inc_Q(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

gamma_inc_Q_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

gammainv(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

gammainv_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

gammastar(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

gammastar_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

gegenpoly_1(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

gegenpoly_1_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

gegenpoly_2(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

gegenpoly_2_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

gegenpoly_3(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

gegenpoly_3_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

gegenpoly_n(...)
Number of Input Arguments: 3 Number of Output Arguments: 1

gegenpoly_n_e(...)
Number of Input Arguments: 3 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 3 of the C argument list

hydrogenicR(...)
Number of Input Arguments: 4 Number of Output Arguments: 1

hydrogenicR_1(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

hydrogenicR_1_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

hydrogenicR_e(...)
Number of Input Arguments: 4 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 4 of the C argument list

hyperg_0F1(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

hyperg_0F1_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

hyperg_1F1(...)
Number of Input Arguments: 3 Number of Output Arguments: 1

hyperg_1F1_e(...)
Number of Input Arguments: 3 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 3 of the C argument list

hyperg_1F1_int(...)
Number of Input Arguments: 3 Number of Output Arguments: 1

hyperg_1F1_int_e(...)
Number of Input Arguments: 3 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 3 of the C argument list

hyperg_2F0(...)
Number of Input Arguments: 3 Number of Output Arguments: 1

hyperg_2F0_e(...)
Number of Input Arguments: 3 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 3 of the C argument list

hyperg_2F1(...)
Number of Input Arguments: 4 Number of Output Arguments: 1

hyperg_2F1_conj(...)
Number of Input Arguments: 4 Number of Output Arguments: 1

hyperg_2F1_conj_e(...)
Number of Input Arguments: 4 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 4 of the C argument list

hyperg_2F1_conj_renorm(...)
Number of Input Arguments: 4 Number of Output Arguments: 1

hyperg_2F1_conj_renorm_e(...)
Number of Input Arguments: 4 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 4 of the C argument list

hyperg_2F1_e(...)
Number of Input Arguments: 4 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 4 of the C argument list

hyperg_2F1_renorm(...)
Number of Input Arguments: 4 Number of Output Arguments: 1

hyperg_2F1_renorm_e(...)
Number of Input Arguments: 4 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 4 of the C argument list

hyperg_U(...)
Number of Input Arguments: 3 Number of Output Arguments: 1

hyperg_U_e(...)
Number of Input Arguments: 3 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 3 of the C argument list

hyperg_U_e10_e(...)
Number of Input Arguments: 3 Number of Output Arguments: 3

The error flag is discarded. Return Arguments 1 - 3 resemble a gsl_result_e10 argument, which is argument 3 of the C argument list

hyperg_U_int(...)
Number of Input Arguments: 3 Number of Output Arguments: 1

hyperg_U_int_e(...)
Number of Input Arguments: 3 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 3 of the C argument list

hyperg_U_int_e10_e(...)
Number of Input Arguments: 3 Number of Output Arguments: 3

The error flag is discarded. Return Arguments 1 - 3 resemble a gsl_result_e10 argument, which is argument 3 of the C argument list

hypot(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

hypot_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

hzeta(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

hzeta_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

laguerre_1(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

laguerre_1_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

laguerre_2(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

laguerre_2_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

laguerre_3(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

laguerre_3_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

laguerre_n (...)
Number of Input Arguments: 3 Number of Output Arguments: 1

laguerre_n_e (...)
Number of Input Arguments: 3 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 3 of the C argument list

lambert_W0 (...)
Number of Input Arguments: 1 Number of Output Arguments: 1

lambert_W0_e (...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

lambert_Wm1 (...)
Number of Input Arguments: 1 Number of Output Arguments: 1

lambert_Wm1_e (...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

legendre_H3d (...)
Number of Input Arguments: 3 Number of Output Arguments: 1

legendre_H3d_0 (...)
Number of Input Arguments: 2 Number of Output Arguments: 1

legendre_H3d_0_e (...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

legendre_H3d_1 (...)
Number of Input Arguments: 2 Number of Output Arguments: 1

legendre_H3d_1_e (...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

legendre_H3d_e (...)
Number of Input Arguments: 3 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 3 of the C argument list

legendre_P1 (...)
Number of Input Arguments: 1 Number of Output Arguments: 1

legendre_P1_e (...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

legendre_P2 (...)
Number of Input Arguments: 1 Number of Output Arguments: 1

legendre_P2_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

legendre_P3(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

legendre_P3_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

legendre_Pl(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

legendre_Pl_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

legendre_Plm(...)
Number of Input Arguments: 3 Number of Output Arguments: 1

legendre_Plm_e(...)
Number of Input Arguments: 3 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 3 of the C argument list

legendre_Q0(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

legendre_Q0_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

legendre_Q1(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

legendre_Q1_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

legendre_Ql(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

legendre_Ql_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

legendre_sphPlm(...)
Number of Input Arguments: 3 Number of Output Arguments: 1

legendre_sphPlm_e(...)
Number of Input Arguments: 3 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 3 of the C argument list

lnbeta (...)
Number of Input Arguments: 2 Number of Output Arguments: 1

lnbeta_e (...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

lnchoose (...)
Number of Input Arguments: 2 Number of Output Arguments: 1

lnchoose_e (...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

lncosh (...)
Number of Input Arguments: 1 Number of Output Arguments: 1

lncosh_e (...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

lndoublefact (...)
Number of Input Arguments: 1 Number of Output Arguments: 1

lndoublefact_e (...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

lnfact (...)
Number of Input Arguments: 1 Number of Output Arguments: 1

lnfact_e (...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

lngamma (...)
Number of Input Arguments: 1 Number of Output Arguments: 1

lngamma_e (...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

lngamma_sgn_e (...)
Number of Input Arguments: 1 Number of Output Arguments: 3

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

lnpoch (...)
Number of Input Arguments: 2 Number of Output Arguments: 1

lnpoch_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

lnpoch_sgn_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 3

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

lnsinh(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

lnsinh_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

log(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

log_1plusx(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

log_1plusx_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

log_1plusx_mx(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

log_1plusx_mx_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

log_abs(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

log_abs_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

log_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

log_erfc(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

log_erfc_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

multiply(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

multiply_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

multiply_err_e(...)
Number of Input Arguments: 4 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 4 of the C argument list

poch(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

poch_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

pochrel(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

pochrel_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

polar_to_rect(...)

pow_int(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

pow_int_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

psi(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

psi_1_int(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

psi_1_int_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

psi_1piy(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

psi_1piy_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

psi_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

psi_int(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

psi_int_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

psi_n(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

psi_n_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

rect_to_polar(...)

sin(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

sin_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

sin_err_e(...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

sinc(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

sinc_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

synchrotron_1(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

synchrotron_1_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

synchrotron_2(...)
Number of Input Arguments: 1 Number of Output Arguments: 1

synchrotron_2_e(...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

taylorcoeff(...)
Number of Input Arguments: 2 Number of Output Arguments: 1

taylorcoeff_e (...)
Number of Input Arguments: 2 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 2 of the C argument list

transport_2 (...)
Number of Input Arguments: 1 Number of Output Arguments: 1

transport_2_e (...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

transport_3 (...)
Number of Input Arguments: 1 Number of Output Arguments: 1

transport_3_e (...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

transport_4 (...)
Number of Input Arguments: 1 Number of Output Arguments: 1

transport_4_e (...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

transport_5 (...)
Number of Input Arguments: 1 Number of Output Arguments: 1

transport_5_e (...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

zeta (...)
Number of Input Arguments: 1 Number of Output Arguments: 1

zeta_e (...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

zeta_int (...)
Number of Input Arguments: 1 Number of Output Arguments: 1

zeta_int_e (...)
Number of Input Arguments: 1 Number of Output Arguments: 2

The error flag is discarded. Return Arguments 1 and 2 resemble a gsl_result argument, which is argument 1 of the C argument list

12.3 Ordinary Functions

The following array functions have been wrapped. These are supposingly faster than the equivalent functions from above.

bessel_In_array(...)

bessel_Jn_array(...)

bessel_Kn_array(...)

bessel_Kn_scaled_array(...)

bessel_Yn_array(...)

bessel_il_scaled_array(...)

bessel_jl_array(...)

bessel_jl_steed_array(...)

bessel_kl_scaled_array(...)

bessel_yl_array(...)

12.4 Ordinary Functions

The following array functions have been wrapped. These are supposingly faster than the equivalent functions from above.

bessel_In_array(...)

bessel_Jn_array(...)

bessel_Kn_array(...)

bessel_Kn_scaled_array(...)

bessel_Yn_array(...)

bessel_il_scaled_array(...)

bessel_jl_array(...)

bessel_jl_steed_array(...)

bessel_kl_scaled_array(...)

bessel_yl_array(...)

APPENDIX

A

pygsl.ieee
Floating Point Unit Support

This chapter lists features to configure the "Floating Point Unit" of your machine. The exact behaviour of your Floating Point Unit can't be discussed here in general — its just machine type dependent.

set_mode(*int precision, int rounding, int exception_mask*)

the mode has effect on the behaviour during calcualtion, e.g. division by zero or rounding.

The following constants are used as precision argument:

mode value	definition via gsl
single_precision	GSL_IEEE_SINGLE_PRECISION
double_precision	GSL_IEEE_DOUBLE_PRECISION
extended_precision	GSL_IEEE_EXTENDED_PRECISION

Possible round arguments are:

mode value	definition via gsl
round_to_nearest	GSL_IEEE_ROUND_TO_NEAREST
round_down	GSL_IEEE_ROUND_DOWN
round_up	GSL_IEEE_ROUND_UP
round_to_zero	GSL_IEEE_ROUND_TO_ZERO

These exception arguments can be added. mask_all is the sum of all 5 mask_* constants.

mode value	definition via gsl
mask_invalid	GSL_IEEE_MASK_INVALID
mask_denormalized	GSL_IEEE_MASK_DENORMALIZED
mask_division_by_zero	GSL_IEEE_MASK_DIVISION_BY_ZERO
mask_overflow	GSL_IEEE_MASK_OVERFLOW
mask_underflow	GSL_IEEE_MASK_UNDERFLOW
mask_all	GSL_IEEE_MASK_ALL
trap_inexact	GSL_IEEE_TRAP_INEXACT

env_setup()

sets the ieee mode from GSL_IEEE_MODE. This is not called any more automatically when importing the pygsl.

bin_repr(*float value*)

returns the binary representation as tuple with the following contents: (int sign, string mantissa, int exponent, int type) These values are used as type in bin_repr:

type value	definition via gsl
type_nan	GSL_IEEE_TYPE_NAN
type_inf	GSL_IEEE_TYPE_INF
type_normal	GSL_IEEE_TYPE_NORMAL
type_denormal	GSL_IEEE_TYPE_DENORMAL
type_zero	GSL_IEEE_TYPE_ZERO

isnan (*float value*)
determines if the argument is not a valid number

nan()
generates a "not-a-number" value. This is implemented as function, because of the potential exception generation by your floating-point unit.

isinf (*float value*)
returns -1 if the argument represents a negative infinite value and +1 if positive, 0 otherwise

posinf()
gives you the representation of "positive infinity"

neginf()
the same as posinf, but negative

finite (*float value*)
results in 1 if the value is finite, 0 if it is not a number or infinite

APPENDIX

B

GNU Free Documentation License

Version 1.1, March 2000

Preamble

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

B.1 Applicability and Definitions

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sec-

tions, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

B.2 Verbatim Copying

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

B.3 Copying in Quantity

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until

at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

B.4 Modifications

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
- State on the Title page the name of the publisher of the Modified Version, as the publisher.
- Preserve all the copyright notices of the Document.
- Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- Include an unaltered copy of this License.
- Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.
- Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties – for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

B.5 Combining Documents

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

B.6 Collections of Documents

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

B.7 Aggregation With Independent Works

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

B.8 Translation

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

B.9 Termination

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

B.10 Future Revisions of This License

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation 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. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

INDEX

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

P

Q

R

S

T

U

V

W

X

Y

Z