PhyPraKit Documentation

Release 1.0

Günter Quast

October 30, 2016

CONTENTS

Übersicht:

PhyPraKit ist eine Sammlung von Funktionen in der Sprache *python (vers. 2.7)* zur Visualisierung und Auswertung von Daten in den physikalischen Praktika. Beispiele illustrieren jeweils die Anwendung.

Version der Dokumentation vom Okt. 2016

CHAPTER

ONE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

CHAPTER

TWO

DARSTELLUNG UND AUSWERTUNG VON MESSDATEN

In allen Praktika zur Physik werden Methoden zur Darstelllung und Auswertung von Messdaten benötigt. Die Script- und Programmiersprache *python* mit den Zusatzpaketen *numpy* und *matplotlib* ist ein universelles Werkzeug, um die Wiederholbarkeit und Reprodzierbarkeit von Datenauswertungen zu gewährleiseten.

In der Veranstaltung "Computergestützte Datenauswertung" (http://www.ekp.kit.edu/~quast/CgDA), die im neuen Studienplan für den Bachelorstudiengang Physik am KIT seit dem Sommersemester 2016 angeboten wird, werden Methoden und Software zur grafischen Darstellung von Daten, deren Modellierung und Auswertung eingeführt.

Die folgen Links erlauben einen schnellen Überblick:

- **Zusammenfassung der Vorlesung und Dokumentation der Code-Beispiele**
  http://www.ekp.kit.edu/~quast/CgDA/CgDA-html/CgDA_ZusFas.html
- **Installation der Software auf verschiedenen Platformen**
  - Dokumentation in html: http://www.ekp.kit.edu/~quast/CgDA/CgDA-SoftwareInstallation-html
  - Dokumentation in pdf: http://www.ekp.kit.edu/~quast/CgDA/CgDA-SoftwareInstallation.pdf
  - Softwarepakete: http://www.ekp.kit.edu/~quast/CgDA/Software

Speziell für das "Praktikum zur klassischen Physik" finden sich eine kurze Einführung (http://www.ekp.kit.edu/~quast/CgDA/PhysPrakt/CgDA_APraktikum.pdf) sowie die hier dokumentierten einfachen Beispiele als Startpunkt für eigene Auswertungen (http://www.ekp.kit.edu/~quast/CgDA/PhysPrakt/).

CHAPTER

THREE

DOKUMENTATION DER BEISPIELE

***PhyPraKit.py* ist ein Paket mit nützlichen Hilfsfunktionen**

zum import in eigene Beispielen mittels:

```
import PhyPraKit as ppk
from PhyPraKit import ...
```

- *test_readColumnData.py* ist ein Beispiel zum Einlesen von Spalten aus Textdateien; die zugehörigen *Metadaten* können ebenfalls an das Script übergeben werden und stehen so bei der Auswsertung zur Verfügung.
- *test_kRegression.py* dient zur Anpassung einer Geraden an Messdaten mit Fehlern in Ordinaten- und Abszissenrichtung und mit allen Messpunkten gemeinsamen (d. h. korrelierten) relativen oder absoluten systematischen Fehlern mit dem Paket *kafe*.
- *test_linRegression.py* ist eine einfachere Version mit *python*-Bordmitteln zur Anpassung einer Geraden an Messdaten mit Fehlern in Ordinaten- und Abszissenrichtung. Korrelierte Unsicherheiten werden nicht unterstützt.
- *test_kFit.py* ist eine verallgemeinerte Version von *test_kRegression* und dient zur Anpassung einer beliebigen Funktion an Messdaten mit Fehlern in Ordinaten- und Abszissenrichtung und mit allen Messpunkten gemeinsamen (d. h. korrelierten) relativen oder absoluten systematischen Fehlern mit dem Paket *kafe*.
- *test_Histogram.py* ist ein Beispiel zur Darstellung und statistischen Auswertung von Häufigkeitsverteilungen (Histogrammen) in einer und zwei Dimensionen.
- *test_generateXYata.py* zeigt, wie man mit Hilfe von Zufallszahlen "künstliche Daten" zur Veranschaulichung oder zum Test von Methoden zur Datenauswertung erzeugen kann.
- *kfitf.py* ist ein Kommandozeilen-Werkzeug, mit dem man komfortabel Anpassungen ausführen kann, bei denen Daten und Fit-Funktion in einer einzigen Datei angegeben werden. Beispiele finden sich in den Dateien mit der Endung *.fit*.

Die übrigen *python*-Scripte im Verzeichnis wurden zur Erstellung der in der einführenden Vorlesung gezeigten Grafiken verwendet.

Für die **Erstellung von Protokollen** mit Tabellen, Grafiken und Formeln bietet sich das Textsatz-System *LaTeX* an. Die Datei *Protokollvorlage.zip* enthält eine sehr einfach gehaltene Vorlage, die für eigene Protokolle verwendet werden kann. Eine sehr viel umfangreichere Einführung sowie ein ausführliches Beispiel bietet die Fachschaft Physik unter dem Link https://fachschaft.physik.kit.edu/drupal/content/latex-vorlagen

CHAPTER

FOUR

MODUL-DOKUMENTATION

Note:

functions contained:

1. Data input from general text files

   read_data()

2. statistics

   wmean() weighted mean

3. histograms tools

   barstat() statistical information bar chart

   nhist() histogram plot based on np.historgram() and plt.bar() `use matplotlib.pyplot.hist() instead`

   histstat() statistical information from 1d-histogram

   nhist2d() 2d-histotram plot based on np.histrogram2d, plt.colormesh() `use matplotlib.pyplot.hist2d() instead`

   hist2dstat() statistical information from 1d-histogram

   profile2d() "profile plot" for 2d data

   chi2p_indep2d() chi2 test on independence of data

4. linear regression

   linRegression() linear regression, y=ax+b, with analytical formula

   linRegressionXY() linear regression, y=ax+b, with x and y errors

   kRegression() regression, y=ax+b. with x-, y- and correlated errors

   kFit() fit function with x-, y- and correlated errors

5. simulated data with MC-method

   smearData() add random deviations to input data

   generateXYdata() generate simulated data

PhyPraKit.**barstat**(*bincont*, *bincent*, *pr=True*)

statistics from a bar chart (histogram) with given bin contents and bin centres

Args:

- bincont: array with bin content

- bincent: array with bin centres

Returns:

- float: mean, sigma and sigma on mean

PhyPraKit.**chi2p_indep2d**(*H2d*, *bcx*, *bcy*, *pr=True*)
perform a chi2-test on independence of x and y

Args:

- H2d: histogram array (as returned by histogram2d)
- bcx: bin contents x
- bcy: bin contents y

Returns:

- float: p-value w.r.t. assumption of independence

PhyPraKit.**generateXYdata**(*xdata*, *model*, *sx*, *sy*, *srelx=None*, *srely=None*, *xabscor=None*, *yabscor=None*, *xrelcor=None*, *yrelcor=None*)
Generate measurement data according to some model assumes xdata is measured within the given uncertainties; the model function is evaluated at the assumed "true" values xtrue, and a sample of simulated measurements is obtained by adding random deviations according to the uncertainties given as arguments.

Args:

- xdata: np-array, x-data (independent data)
- model: function that returns (true) model data (y-dat) for input x

the following are single floats or arrays of length of x

- sx: gaussian uncertainty(ies) on x
- sy: gaussian uncertainty(ies) on y
- srelx: relative gaussian uncertainty(ies) on x
- srely: relative gaussian uncertainty(ies) on y

the following are common (correlated) systematic uncertainties

- xabscor: absolute, correlated error on x
- yabscor: absolute, correlated error on y
- xrelcor: relative, correlated error on x
- yrelcor: relative, correlated error on y

Returns:

- np-arrays of floats:
  - xtrue: true x-values
  - ytrue: true value = model(xtrue)
  - ydata: simulated data

PhyPraKit.**hist2dstat**(*H2d*, *xed*, *yed*, *pr=True*)
calculate statistical information from 2d Histogram

Args:

- H2d: histogram array (as returned by histogram2d)

- xed: bin edges in x
- yed: bin edges in y

Returns:

- float: mean x
- float: mean y
- float: variance x
- float: variance y
- float: covariance of x and y
- float: correlation of x and y

PhyPraKit.**histstat**(*binc*, *bine*, *pr=True*)

calculate mean of a histogram with bincontents binc and bin edges bine

Args:

- binc: array with bin content
- bine: array with bin edges

Returns:

- float: mean and sigma

PhyPraKit.**kFit**(*func, x, y, sx, sy, xabscor=None, yabscor=None, xrelcor=None, yrelcor=None, title='Daten', axis_labels=['X', 'Y'], plot=True, quiet=False*)

fit function func with errors on x and y; uses package *kafe*

Args:

- func: function to fit
- x: np-array, independent data
- y: np-array, dependent data

the following are single floats or arrays of length of x

- sx: uncertainty(ies) on x
- sy: uncertainty(ies) on y
- xabscor: absolute, correlated error(s) on x
- yabscor: absolute, correlated error(s) on y
- xrelcor: relative, correlated error(s) on x
- yrelcor: relative, correlated error(s) on y
- title: string, title of gaph
- axis_labels: List of strings, axis labels x and y
- plot: flag to switch off graphical ouput
- quiet: flag to suppress text and log output

Returns:

- np-array of float: parameter values
- np-array of float: parameter errors

- np-array: cor correlation matrix
- float: chi2 chi-square

PhyPraKit.**kRegression**(*x, y, sx, sy, xabscor=None, yabscor=None, xrelcor=None, yrelcor=None, title='Daten', axis_labels=['X', 'Y'], plot=True, quiet=False*)

linear regression y(x) = ax + b with errors on x and y; uses package *kafe*

Args:

- x: np-array, independent data
- y: np-array, dependent data

the following are single floats or arrays of length of x

- sx: uncertainty(ies) on x
- sy: uncertainty(ies) on y
- xabscor: absolute, correlated error(s) on x
- yabscor: absolute, correlated error(s) on y
- xrelcor: relative, correlated error(s) on x
- yrelcor: relative, correlated error(s) on y
- title: string, title of gaph
- axis_labels: List of strings, axis labels x and y
- plot: flag to switch off graphical ouput
- quiet: flag to suppress text and log output

Returns:

- float: a slope
- float: b constant
- float: sa sigma on slope
- float: sb sigma on constant
- float: cor correlation
- float: chi2 chi-square

PhyPraKit.**linRegression**(*x, y, sy*)

linear regression y(x) = ax + b

Args:

- x: np-array, independent data
- y: np-array, dependent data
- sx: np-array, uncertainty on y

Returns:

- float: a slope
- float: b constant
- float: sa sigma on slope

- float: sb sigma on constant
- float: cor correlation
- float: chi2 chi-square

PhyPraKit.**linRegressionXY**(*x*, *y*, *sx*, *sy*)
linear regression y(x) = ax + b with errors on x and y uses numerical "orthogonal distance regression" from package scipy.odr

Args:

- x: np-array, independent data
- y: np-array, dependent data
- sx: np-array, uncertainty on y
- sy: np-array, uncertainty on y

Returns:

- float: a slope
- float: b constant
- float: sa sigma on slope
- float: sb sigma on constant
- float: cor correlation
- float: chi2 chi-square

PhyPraKit.**nhist**(*data*, *bins=50*, *xlabel='x'*, *ylabel='frequency'*)
Histogram.hist show a one-dimensional histogram

Args:

- data: array containing float values to be histogrammed
- bins: number of bins
- xlabel: label for x-axis
- ylabel: label for y axix

Returns:

- float arrays bin content and bin edges

PhyPraKit.**nhist2d**(*x*, *y*, *bins=10*, *xlabel='x axis'*, *ylabel='y axis'*, *clabel='counts'*)
Histrogram.hist2d create and plot a 2-dimensional histogram

Args:

- x: array containing x values to be histogrammed
- y: array containing y values to be histogrammed
- bins: number of bins
- xlabel: label for x-axis
- ylabel: label for y axix
- clabel: label for colour index

Returns:

- float array: array with counts per bin
- float array: histogram edges in x
- float array: histogram edges in y

PhyPraKit.**profile2d**(*H2d*, *xed*, *yed*)

generate a profile plot from 2d histogram:

- mean y at a centre of x-bins, standard deviations as error bars

Args:

- H2d: histogram array (as returned by histogram2d)
- xed: bin edges in x
- yed: bin edges in y

Returns:

- float: array of bin centres in x
- float: array mean
- float: array rms
- float: array sigma on mean

PhyPraKit.**readColumnData**(*fname*, *ncols=4*, *cchar='#'*, *delimiter=None*, *pr=True*)

read column-data from file

Args:

- string fnam: file name
- int ncols: number of columns
- char delimiter: character separating columns
- bool pr: print input to std out if True

PhyPraKit.**smearData**(*d*, *s*, *srel=None*, *abscor=None*, *relcor=None*)

Generate measurement data from "true" input d by adding random deviations according to the uncertainties

Args:

- d: np-array, (true) input data

the following are single floats or arrays of length of array d

- s: gaussian uncertainty(ies) (absolute)
- srel: gaussian uncertainties (relative)

the following are common (correlated) systematic uncertainties

- abscor: absolute, correlated uncertainty
- relcor: relative, correlated uncertainty

Returns:

- np-array of floats: dm, smeared (=measured) data

PhyPraKit.**wmean**(*x*, *sx*, *pr=True*)

weighted mean of np-array x with uncertainties sx

Args:

- x: np-array of values
- sx: uncertainties
- pr: if True, print result

Returns:

- float: mean, sigma

test_readColumnData.py

test data input from text file with module PhyPraKit.readColumnData

test_Historgram.py

test histogram functionality in PhyPraKit

test_kRegression test linear regression with kafe, uncertainties in x and y and correlated absolute and relative uncertainties

test_kFit test fiting an arbitrary fucntion with kafe, with uncertainties in x and y and correlated absolute and relative uncertainties

test_generateDate

test generation of simulated data this simulates a measurement with given x-values with uncertainties; random deviations are then added to arrive at the true values, from which the true y-values are then calculated according to a model function. In the last step, these true y-values are smeared by adding random deviations to obtain a sample of measured values

kfitf.py

Perform a fit with the kafe package driven by input file

usage: kfitf.py [-h] [-n] [-s] [-c] [–noinfo] [-f FORMAT] filename

positional arguments: filename name of fit input file

optional arguments:

-h, --help	show this help message and exit
-n, --noplot	suppress ouput of plots on screen
-s, --saveplot	save plot(s) in file(s)
-c, --contour	plot contours and profiles
--noinfo	suppress fit info on plot
--noband	suppress 1-sigma band around function
--format FMT	graphics output format, default FMT = pdf

k

p

t