Package 'kzfs'

June 2, 2019

Title Multi-Scale Motions Separation with Kolmogorov-Zurbenko Periodogram Signals

Version 1.5.0.2

Author Ming Luo <ml226662@gmail.com> and Igor Zurbenko <IZurbenko@albany.edu>

Maintainer Ming Luo <ml226662@gmail.com>

Description Separation of wave motions in different scales and directions based on Kolmogorov-Zurbenko Periodograms and Kolmogorov-Zurbenko Fourier Transform.

Depends R (>= 3.3.2)

License GPL (>= 2)

LazyData true

Imports methods, digest, kzft

RoxygenNote 6.1.1

Suggests polynom

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-06-02 16:20:03 UTC

R topics documented:

`kzfs` *Multi-Scale Motion Separation with Kolmogorov-Zurbenko Periodogram Signals*

Description

Motion image identification in different types of data is very important subject in many applications. Those images may depend on time and contain different scales. A simple example is waves in the ocean coming from two different directions. One wave can be strong long scale, and another is shorter scale wave propagating in different direction. When both are covered by strong noise, data realization could be very noisy 3D structure. Similar examples can be presented in engineering, acoustics, astronomy, infection diseases developments and many other fields.

This package is designed for the separation of motion scales in 2D motion images on different directions. To this end, KZ periodogram is utilized to identify spatial directions and frequencies of wave signals, while KZ Fourier transform provides the reconstructed signals based on identified motion parameters.

By spectral analysis of original signal in different directions, we can discover main directions in which different scale waves are propagating. Intuitively, sampling along the orthogonal direction of a wave will annihilate its frequency spike on the corresponding periodogram. Therefore, the presence and absence of single frequency on the periodograms of different directions can be used to identify the wave direction. This method can be enriched by finding the common projected spectral spikes detected from a series of periodograms for different sampling directions. Identification of wave frequencies can be done symmetrically.

For the task of identification, this package provides functions to check averaged periodogram for data series in a given direction or a group of directions. Averaging of these directional periodograms will help to stable the variance of spectrum. Functions are provided for automatically identifying and marking prominent spectrum spikes. The closure of nearest-neighbors is used to detect the clusters formed by real waves on the frequency-direction plane. The algorithm is designed to resist incorrectly identified periodogram signals caused by noises, and it gives consistent estimations when the number of sampling directions increases. The accuracy of the estimations can also be improved with the increase of the sampling number.

In the stage of signal reconstruction, Fourier transform is utilized as a powerful tool to recover signals series. kzfs package provides function to reconstruct 2D spatial waves under noisy background. Reconstructed signal can be averaged along the vertical lines of its propagating direction. This will significantly reduce the noise effects and improve the accuracy of reconstruction.

kzfs also provides functions to improve the estimation accuracy of wave parameters with optimization on KZ directional periodograms and 2D periodograms. The optimized wave parameter estimations will improve the accuracy of reconstruction with Fourier transform. This is especially useful in cases of relative short data series and small window sizes.

References

- I. G. Zurbenko, The spectral Analysis of Time Series. North-Holland, 1986.

- A. G. DiRienzo, I. G. Zurbenko, Semi-adaptive nonparametric spectral estimation, Journal of Computational and Graphical Statistics 8(1): 41-59, 1998.
- R. Neagu, I. G. Zurbenko, Algorithm for adaptively smoothing the log-periodogram, Journal of the Franklin Institute 340: 103-123, 2003.
- I. G. Zurbenko, M. Luo, Restoration of Time-Spatial Scales in Global Temperature Data, American Journal of Climate Change, 1(3): 154-163, 2012.
- I. G. Zurbenko, M. Luo, Surface Humidity Changes in Different Temporal Scales, America Journal of Climate Change, 4(3): 226-238, 2015.
- M. Luo, I. G. Zurbenko, KZ Spatial Waves Separations, Journal of Research in Applied Mathematics, 3(4):1-7, 2017.
- M. Luo, I. G. Zurbenko, Spectral Feature of Sampling Errors for Directional Samples on Gridded Wave Field, International Journal of Engineering Research and Technology, 5(12):525-531, 2016.

See Also

kzpdr, kzp2, kzrc2

`kzft` *Kolmogorov-Zurbenko Fourier Transform Function*

Description

`kz.ft` is improved version of Wei Yang's `kzft::kzft`. It has been modified to handle missing values in the data. Besides KZ Fourier transform, the outputs also include KZ periodogram.

`kz.ftc` is an experimental version of KZFT for signals sampled on continual time/space points with irregular intervals. Missing is common in this scheme. However, you may need large window size for reconstruction of the signals.

Usage

```
kz.ft(x, m, ...)

kz.ftc(x, m, ...)
```

Arguments

`x`	The data vector. Missing values are allowed.
`m`	The window size for a regular Fourier transform
`...`	Other arguments.

- `k` : Integer. The iterations number of KZFT.
- `f` : Vector. Selected frequencies. Default value is c(1:m)/m
- `n` : The sampling frequency rate as a multiplication of the Fourier frequencies

- `p` : The distance between two successive intervals as a percentage of the total length of the data series.
- `adpt` : Logic. Flag for using adaptive window size, or not. Default is FALSE.
- `phase` : Logic. Flag for correcting phase shift, or not. Default is FALSE.

Details

If *2*m*f* is not an integer, the recovered signal may have included a phase shift. However, if the option of "phase shift correction is enabled, the related errors caused by the phase shift can be limited to an acceptable level. Please notice that this method is useful for cases with low background noise and the aim is to near perfectly recover the signal. The targeted signal also needs to be the dominant signal in the data so that the interaction of other signals is negligible.

Another way to reduce the errors caused by unmatched m and f is to use the option of "adaptive window size". It will help you select the best window size for the given frequencies and the data length automatically. But it only works well when it exists m for integer values *2*m*f*.

These two options haven't been implemented for `kz.ftc`.

Value

List. It includes data frame for Fourier transform matrix `tfmatrix`, column means of Fourier transform matrix `fft`, vector `pg` and `f` for KZ-periodogram values and corresponding frequencies.

See Also

`kzft`, `kz.smpg`, `kzp2`

Examples

```
## Adapted from kzft::kzp example 2
t <- 1:2000
y <- 1.1*sin(2*pi*0.0339*t)+7*sin(2*pi*0.0366*t)
y2 <- y
noise <- rnorm(length(t),0,1)
y[sample(t,100,replace=FALSE)] <- NA
f <- c(0.0339, 0.0366)

## Periodogram
ft <- kz.ft(y+5*noise, f=f, k=2, m=1000, n=10)
# It may take 10 ~ 20 seconds
# system.time(ft <- kz.ft(y+5*noise, k=2, m=1000, n=10))
plot(y=log(ft$pg+1), x=ft$f, type="l", xlim=c(0.025,0.045))
abline(v=f, lty=21, col="red")
text(x=f+0.001, y=c(2,4), f, col="red", cex=0.75)

## recover signal
ft <- kz.ft(y+5*noise, f=f, k=3, m=500)
yr <- 2*Re(rowSums(ft$tf))
cor(yr, y2[1:length(yr)], use="pairwise.complete.obs")
plot((y+5*noise)[1:length(yr)], type="p", cex=0.5, col="grey")
```

```
points(y[1:length(yr)],type="b", col="red", cex=0.45)
points(yr, type="p", cex=0.35, col="blue")
mtext("Red dots: singal, Blue dots: reconstruction", cex=0.75)


## Additional example
t <- 1:2000
y <- 1.1*sin(2*pi*0.011*t)+2*sin(2*pi*0.032*t)
y2 <- y
y[sample(t,500,replace=FALSE)] <- NA
noise <- rnorm(2000,0,1)
ft <- kz.ft(y + 3.0*noise, k=5, f=c(0.011,0.032), m=300, adpt=FALSE)
yr <- 2*Re(rowSums(ft$tf))
cor(yr, y2[1:length(yr)], use="pairwise.complete.obs")
plot((y+5*noise)[1:length(yr)], type="p", col="grey")
points(y2[1:length(yr)], type="l", col="red")
points(y[1:length(yr)],type="p", col="red", cex=0.35)
points(yr, type="p", cex=0.3, col="blue")
mtext("Red: singal, Grey: singal + 5*noise, Blue: reconstruction", cex=0.75)


## Example for kz.ftc

t <- runif(2000)*2000
f <- c(0.15, 0.1)
x <- sin(2*pi*f[1]*t + pi/4)
y <- sin(2*pi*f[2]*t + pi/12)
y <- y[order(t)]
x <- x[order(t)]
tr <- t[order(t)]
noise <- rnorm(length(tr),0,1)
plot(y=y+x, x=tr, type="l")

## Periodogram
ft <- kz.ftc(x+y+2*noise, xt=tr, k=2, m=1000)
plot(y=ft$pg, x=ft$f, type="l")
abline(v=f, col="grey", lty=21)
text(x=f+0.001, y=c(200,400), f, col="red", cex=0.75)
mtext("Spectrum of Longitudinal Data, Selected f")

## recover signal
ft <- kz.ftc(x+y+noise, xt=tr, f=f, k=1, m=1900)
yr <- rowSums(2*Re(ft$tf))
iv <- 0:60
plot(y=(x+y+noise)[iv], x=tr[iv], type="p", col="grey")
xt <- (0:8000)/100
yt <- sin(2*pi*f[1]*xt+pi/4) + sin(2*pi*f[2]*xt+pi/12)
y2 <- sin(2*pi*f[1]*iv+pi/4) + sin(2*pi*f[2]*iv+pi/12)
points(yt, x=xt, col="grey", cex=0.5, lwd=1, type="l")
points(y2, x=iv, col="blue", cex=0.75, lwd=1, type="p")
points(y=yr, x=0:(length(yr)-1), type="p", cex=0.5, lwd=1, col="red")
mtext("Red: reconstruction, Grey: signal + noise", cex=0.75)
```

kzmd *Yet Another Multi-dimensional Kolmogorov-Zurbenko Filter*

Description

This implement of spatial KZ-filter works for any dimensions. It is designed for cases with sparse data in large time-space.

Usage

```
kzmd(ss, window, scale, k = 1, edges = TRUE)
```

Arguments

ss	Data frame with value column behind time/space coordinates.
window	Vector for window size of each dimension.
scale	Vector for scale of each dimension.
k	Iteration times of KZ filter. Defaults to 1.
edges	Logic. Defaults to TRUE. FLASE means clear the data that are located outside the time-space range of input data.

Value

Data framework with value column behind time/space coordinates.

See Also

kz

Examples

```
zs <- rbind(c(0,5,1,40),c(12,6,1,10),c(6,7,1,20),c(15,15,4,80))
colnames(zs) <- c("x","y","z","v")
zs <- kzmd(data.frame(zs), scale=c(1,1,1), window=c(3,5,3), k=4)
u <- zs[zs$z==1, -3]
x = sort(unique(u$x))
y = sort(unique(u$y))
z=df2mt(u, scale=c(1,1)) # Transfer from data frame to matrix.
image(x=x, y=y, z=z)
```

kzp2 *Check Images' Motion Scales with 2D KZ Periodogram Signals*

Description

Functions used to reveal directional and scale information with 2D KZ periodograms for spatial motions covered by heavy noises.

One can get 2D raw periodogram with function `kzp2`, and smooth the 2D periodogram with function `smooth.kzp2`.

Function `summary.kzp2` can help to summarize direction and frequency information from smoothed 2D KZ periodogram. The input should be a 2D KZ periodogram data with frequency range (0, 0.5] on both x- and y- axis.

Usage

```
kzp2(x, k = 1, m = dim(x)/k, ...)

smooth.kzp2(rpg, dpct = 0.01, w = round(dim(rpg)/4), k = 1, ...)

kzp2.summary(spg, rg.x, rg.y, num = 10)
```

Arguments

`x`	Data array of 2D wave field. Missing values are allowed. Limited to 2D arrays for current version.
`k`	The number of iterations for the KZFT. Default is 1.
`m`	The window size for a regular Fourier transform. Default value is set to data array size.
`...`	Arguments to be passed to methods. • `k` : The number of iteration times of KZFT • `n` : The sampling frequency rate as a multiplication of the Fourier frequencies • `p` : The distance between two successive intervals as a percentage of the total length of the data series
`rpg`	Array of raw 2D periodogram. Usually it is part of output of `kzp2`.
`dpct`	A pre-specified percentage of total variation. Default value is 1%.
`w`	Smoothing window size.
`spg`	Array of smoothed 2D periodogram. It could be output of `summary.kzp2`.
`rg.x`	Frequency range for x direction. Defaults to c(0, 0.5).
`rg.y`	Frequency range for y direction. Defaults to the same value of the range for x direction.
`num`	Wave numbers. Defaults to 10.

Details

KZ 2D raw spectrum is calculated based on `kz.ft`. The smoothing method is an extension of `kzft::smooth.kzp`. See introduction of DZ method in `kzft::smooth.kzp` for more information.

Value

Returned value of function `kzp2` is a data list of periodogram information, including data array *kzp2d* for 2D periodogram values, and two frequency vectors, *freq.x* and *freq.y* for *x* and *y* direction, respectively.

`smooth.kzp2` only outputs the array of smoothed values.

`kzp2.summary` returns a data list for suggested wave parameters, including frequency and direction values.

See Also

`kzpdr`, `kzpdr.eval`, `kzpdr.spikes`

Examples

```
dx <- 100 # x range
dy <- 120 # y range
b <- expand.grid(x=1:dx, y=1:dy)
q1 <- pi/6; f1 <- 0.2;
b$v1 <- sin(f1*2*pi*(b$x*cos(q1)+b$y*sin(q1))+100*runif(1))
q2 <- pi/4; f2 <- 0.08;
b$v2 <- sin(f2*2*pi*(b$x*cos(q2)+b$y*sin(q2))+100*runif(1))
a <- array(0,c(dx,dy))
a[as.matrix(b[,1:2])] <- b$v1 + 1.5*b$v2
a <- a + 10*matrix(rnorm(dx*dy,0,1),ncol=dy)

rp <- kzp2(a) # raw 2D spectrum

fy <- rp$freq.y; fx <- rp$freq.x; rp <- rp$kzp2d

# smoothing 2D spectrum 2 times
sp <- smooth.kzp2(rp,0.01,k=2)

par(mfrow=c(2,1), cex=0.5)
persp(x=fx, y=fy, z=rp, expand =0.5,
main = "Raw 2D KZ Periodogram", ltheta=40, shade=0.75,
theta=-30, phi=15, zlab="",xlab="x", ylab="y",
ticktype="detailed", col="lightblue")

persp(x=fx, y=fy, z=sp, expand =0.5,
main = "Smoothed 2D KZ Periodogram", ltheta=40, shade=0.75,
theta=-30, phi=25, zlab="",xlab="x", ylab="y",
ticktype="detailed", col="lightblue")
par(mfrow=c(1,1), cex=1)

kzp2.summary(sp) # direction & frequency
```

kzp2.QF — *The Demo Dataset For Examples of* optD2R

Description

Datasets containing the original and optimized 2D periodogram spike records output by function kzp2 and optD2R,respectively. They are only used for saving running-time of the examples. The following code is used to generate this dataset:

- `kzp2.demo <- kzp2(a)$kzp2d`
- `kzp2.QF <- optD2R(a, kzp2.summary(kzp2.demo, num=2))`

where a is the spatial data array for a wave field.

Usage

```
kzp2.demo

kzp2.QF
```

Format

An object of class matrix with 150 rows and 150 columns.

See Also

kzpdr.demo, kzpdr.QF

kzpdr — *Average Periodogram for Spatial Data in Given Directions*

Description

Functions in this group are designed to check periodogram for data series in a given direction or a list of directions.

kzpdr samples the data of wave field, and outputs the average pattern of periodogram for series in a given direction. A collection of these pattern records will be sent to kzpdr.eval or kzpdr.estimate to estimate the wave frequencies and directions.

Usage

```
kzpdr(ds, angle, plot = F, pair = T, ...)

kzpdr.3d(ds, angle, ...)
```

Arguments

ds	Data array. Only 2 dimensional arrays are allowed for current version.
angle	Vector or single numeric value in radians.
plot	TRUE or FLASE. Flag for outputting designed periodogram plot or not. Defaults to FLASE. In kzpdr, the plot is the mean periodogram for data series in a given direction.
pair	Logic. Defaults to TRUE, i.e., check the given directions and their orthogonal opposition at the same time.
...	Other arguments.

- For function kzpdr, it could be the following arguments (right of equals signs are their default setting):
  - w = 20 : smoothing window size.
  - dpct = 0.01 : a percentage of total variation of periodogram; smoothing window is extended until variation within the window reaches this number. See DZ method in kzft::kzp for details.
  - rg = c(0,0.5) : the frequency range for the periodogram.
  - raw = FALSE : if use the raw periodogram directly.
  - log = FALSE : if use log scale for periodogram.
  - frun = FALSE : if force to run the sampling on given directions. Defaults to check records and not sample on duplicate directions
  - min.ln = 0.6 : the minimum ratio of sampling data length vs. original data length to product a periodogram for a direction.
- In kzpdr.3d function, it could be arguments of the perspective plot, like theta, phi, etc., please refer function graphics::persp for more information.
- For kzpdr.valid, level control the cross-validation process: integer number k means to run cross-validation by excluding k pairs of directional samples each time. Default value is 1.

Details

kzpdr is used to sample the spatial data and generates periodograms in orthogonal direction pairs; the frequencies of spikes for each directional periodogram are identified and recorded as the function output. The spike pattern of average periodograms for spatial directions can help to identify wave frequencies and directions.

Function kzpdr.3d will provide 3D perspective plot as the global view for periodograms of data series in a given direction.

Value

The returned data list of function kzpdr includes the data frame for frequencies of spikes on mean periodograms of each checked direction. It also includes a vector recording the md5sum value of the spatial wave data array for internal control.

Function kzpdr will output the periodogram plots when option plot is set as TRUE. The frequencies of marked spikes will also be print out for each sampling direction.

kzpdr.3d returns back the data frame for re-gridded mean periodogram for data series in given direction, as showed in the perspective plot.

See Also

kzp2, kzpdr.tol, kzpdr.eval kzpdr.valid, kzpdr.spikes

Examples

```
dx <- 300
dy <- 300

b <- expand.grid(x=1:dx, y=1:dy)
q1 <- pi/3; f1 <- 0.2;
b$v1 <- sin(f1*2*pi*(b$x*cos(q1)+b$y*sin(q1))+100*runif(1))
q2 <- pi/6; f2 <- 0.05;
b$v2 <- sin(f2*2*pi*(b$x*cos(q2)+b$y*sin(q2))+100*runif(1))

a <- array(0,c(dx,dy))
a[as.matrix(b[,1:2])] <- b$v1 + 1.5*b$v2
persp(1:dx, 1:dy, a, theta=90, phi=-110,
ticktype="detailed", col="lightblue")
a <- a + 5*matrix(rnorm(dx*dy,0,1),ncol=dy)
persp(1:dx, 1:dy, a, theta=90, phi=-110,
ticktype="detailed", col="lightblue")

# It may take about 30 seconds
# o <- kzpdr.3d(a, -pi/6)

# Load pre-saved data to save running-time
data(kzpdr.demo);

# sampling, it may take a few minutes
# system.time(kzpdr.demo <- kzpdr(a, pi/12, pair=FALSE, plot=TRUE))
# system.time(kzpdr.demo <- kzpdr(a, pi/12, plot=TRUE))
# system.time(kzpdr.demo <- kzpdr(a, c(0, pi/6, pi/4, pi/3), plot=TRUE))

kzpdr.spikes(kzpdr.demo)

# For identification of the wave parameters, see kzpdr.estimate
```

---

kzpdr.eval — *Evaluate Directional Spectrum Data for Wave Frequencies and Directions*

Description

Functions in this group are designed to estimate wave parameters based on directional periodogram records.

kzpdr samples the data of wave field, and outputs the average pattern of periodogram for series in a given direction. A collection of these pattern records will be sent to kzpdr.eval or kzpdr.estimate to estimate the wave frequencies and directions.

Usage

```
kzpdr.eval(rec = ls(1), t.D = 2, t.F = 0.01, ...)

kzpdr.estimate(rec = ls(1), ...)
```

Arguments

rec	Data list from the outputs of function kzpdr. It includes the data frame for the marked frequency values and corresponding directions. Defaults is searching for available records in the environment.
t.D	Tolerance of direction in degree. Default is 2.
t.F	Tolerance of frequency. Default value is 0.01.
...	Other arguments. • D3 Logic. Default is FALSE. If TRUE, output 3D perspective plot; otherwise, 2D plot on frequency-direction surface. • scale A two element vector for grid on frequency-direction plant. The first element is for frequency. The second is for degree of direction. Default is c(0.005,1). • ... ...

Details

The average periodograms for a few pairs of orthogonal spatial directions can be used to identify frequencies and directions of waves.

First, function kzpdr samples the spatial data and generates periodograms in orthogonal direction pairs, and the frequencies of spikes for each directional periodogram are identified and recorded as the output.

Then, kzpdr.spikes can be used to summarize the outputs of kzpdr. Function kzpdr.eval or kzpdr.estimate all can be used to estimate the wave parameters (frequencies and directions). kzpdr.estimate is based on clustering-closure and the tolerances could be decided automatically. It also provides visualization of the results, thus this function is more convenient to use.

Usually, if noise level is low, periodograms of a few direction pairs may provide satisfied results. But when the noise is high, you may need to intensively sample on different directions over the spatial data array with kzpdr. Generally speaking, when the number of samples increases, the estimation will become more stable and reliable.

Value

Both kzpdr.eval and kzpdr.estimate will return suggested wave frequency and direction values. The data frame of detailed estimation for each direction is also included in their returned data list. Beside these, kzpdr.estimate can generate 3D or 2D plots for the supports of each suggested wave on direction-frequency parameter plane.

See Also

kzpdr, kzpdr.valid, kzp2 kzpdr.tol, kzpdr.spikes

Examples

```
# load pre-saved data to save running-time
data(kzpdr.demo);

# estimate the wave parameters
kzpdr.eval(kzpdr.demo, t.D = 3, t.F = 0.01)

# estimation & visualization
kzpdr.estimate(kzpdr.demo)

# For validation of the estimation, see \code{kzpdr.valid}
# For reconstruction of the signals, see \code{kzrc}
```

---

kzpdr.QF *The Demo Dataset For Examples of* kzpdr *and* optDR

Description

Datasets containing the spectral spike records of directional periodograms output by function kzpdr and optDR, respectively. They are only used for saving running-time of the examples. The following code is used to generate this dataset:

- `kzpdr.demo <- kzpdr(a, c(0, pi/4, pi/3, -pi/3, pi/18), plot=TRUE)`
- `kzpdr.QF <- optDR(a, kzpdr.demo)`

where a is the spatial data array for a wave field.

Usage

```
kzpdr.demo

kzpdr.QF
```

Format

A list for a data frame and its MD5sum value. The data frame rec has 18 rows and a few variables:

direction angle of sampling, in degree

freq frequency values of spikes

spg smoothed power periodogram values of spikes

... ...

See Also

kzp2.demo, kzp2.QF

kzpdr.spikes *Count Spikes For Available Directional Periodogram Records*

Description

Function kzpdr.spikes summarizes available periodogram pattern records collected from the outputs of kzpdr, gives expected wave number. This number is used by kzpdr.tol and kzpdr.valid in searching feasible tolerance setting and validation of estimated wave parameters.

Usage

```
kzpdr.spikes(rec = ls(1))
```

Arguments

rec	Data list from the outputs of function kzpdr. It includes the data frame for the marked frequency values and corresponding directions. Defaults is searching for available records in the environment.

Details

The expected wave number is defined as the mode of all the spike counts.

If any of the sampling direction in the available directional periodogram records, say A, happens to be orthogonal to a wave direction B, then there will be no spike appear on related periodogram for the wave propagated in direction B. Related spike counts will be less than the expected wave number. The absence of spike(s) in one direction can be taken as the evidence for the existing of wave(s) in its orthogonal direction.

Usually, it is very rare to have a simpling direction orthogonal to a wave direction. But if we know an approximate wave direction, we can take more samplings around its orthogonal direction. Since KZ periodogram can separate wave spikes in very close frequencies, we may get more accurate estimation for this wave direction with this method. It is also possible to use this way to validate estimations get by other approaches.

See Also

kzpdr, kzpdr.eval kzpdr.valid, kzpdr.tol

Examples

```
# load pre-saved data
data(kzpdr.demo);

# count spikes
kzpdr.spikes(kzpdr.demo)
```

kzpdr.tol *Search Appropriate Tolerances Setting for Wave Parameter Estimation*

Description

kzpdr.tol will help to find the feasible tolerance settings for the wave parameter estimation.

Usage

```
kzpdr.tol(rec = ls(1), t.D = seq(1, 10, 1), t.F = 0.01)
```

Arguments

rec	Data list from the outputs of function kzpdr. It includes the data frame for the marked frequency values and corresponding directions. Defaults is searching for available records in the environment.
t.D	Vector for search range of direction tolerance. Default is 1:10 (in degree).
t.F	Vector for search range of frequency tolerance. Default value is c(0.01).

Details

Since the expected wave number is known(see kzpdr.spikes), we can search for the tolerance settings that would generate estimations with wave number in this range. A table will be presented to summary feasible settings.

The searching process would stop when it finsihed the search range, or the increasing of the tolerance led to null result.

See Also

kzpdr, kzpdr.eval kzpdr.valid, kzpdr.spikes

Examples

```
# load pre-saved data
data(kzpdr.demo);

# search for tolerance
kzpdr.tol(kzpdr.demo, t.D = c(1,2,3), t.F = 0.005)
```

`kzpdr.valid` *Validate Estimated Wave Parameters Under Given Tolerance Setting*

Description

For a given tolerance setting, `kzpdr.valid` will provide cross-validation information for related results of wave parameter estimations.

Usage

```
kzpdr.valid(rec = ls(1), t.D = 2, t.F = 0.01, level = 1)
```

Arguments

`rec`	Data list from the outputs of function `kzpdr`. It includes the data frame for the marked frequency values and corresponding directions. Defaults is searching for available records in the environment.
`t.D`	Tolerance of direction in degree. Default is 2.
`t.F`	Tolerance of frequency. Default value is 0.01.
`level`	`level` control the cross-validation process: integer number k means to run cross-validation by excluding k pairs of directional samples each time. Default value is 1.

Details

Due to the nosies or other reasons, there may exist fake spike signals in the directional periodograms. Cross-validation will evaluate estimations by excluding one or few measurements, and identify the data points that casued inconsistent estimations. A table will be given to summarize the validation results.

For a given tolerance setting, if the validation shows consistent results, related estimation would be reliable.

See Also

`kzpdr`, `kzpdr.eval`, `kzpdr.tol`

Examples

```
# load pre-saved data
data(kzpdr.demo)

# validation
kzpdr.valid(kzpdr.demo, t.D = 2, t.F = 0.01, level = 1)
```

kzrc2 *Reconstruct 2D Wave Signals For Given Directions with KZFT*

Description

Once you have identified the waves' directions and frequencies, you can reconstruct the spatial wave signals with `kz.rc2`. Directional information is utilized to suppressive the noise.

Usage

```
kz.rc2(ds, f = 0.25, m = round(min(dim(ds)/2)), ...)
```

Arguments

`ds`	Matrix for data of wave field. Missing values are allowed.
`f`	Vector. Identified wave frequency.
`m`	The window size for a regular Fourier transform
`...`	Other arguments.

- `angle` : Vector. Identified wave direction value in degree.
- `k` : Integer, defaulting to 2. The iterations number of KZFT.
- `n` : The sampling frequency rate as a multiplication of the Fourier frequencies
- `p` : The distance between two successive intervals as a percentage of the total length of the data series
- `avg` : Logic. If average along orthogonal direction. Default is TRUE if `angle` is available.
- `plot` : Logic. Flag for outputing figures. Default is FALSE.
- `compare` : Logic. Flag for drawing input image. Default is FALSE.
- `edge` : Logic. Flag for keeping the edge data in the returned reconstructed signal. Default is FALSE.
- `rlvl` : Integer. Coefficient to control the averaging level when `avg` is TRUE. Default value is 2.

Details

Averaging along the orthogonal direction of a wave signal will significantly reduce the noise effects and increase the accuracy of reconstruction.

When the direction information is not available, the 2D signal will be reconstructed along x-axis, but the result usually has a phase-shift even for the dominant wave pattern.

If choose to average reconstructed signal along its orthogonal direction, `rlvl` should be set to control the averaging level. If the input data has comparable size on x- or y-dimension, an experiential formula is to use the integer value around `sqrt(dx)*1.5`, where `dx` is the array size.

See Also

kzft, kz.smpg, kzp2

Examples

```
dx <- 100 # The x and y scale of the wave field
dy <- 100 # Enlarge them to 300 to get better result.

b <- expand.grid(x=1:dx, y=dy:1)
q1 <- pi/6; f1 <- 0.1;
b$v1 <- sin(f1*2*pi*(b$x*cos(q1)+b$y*sin(q1))+runif(1))
a1 <- array(0,c(dx,dy))
a1[as.matrix(b[,1:2])] <- b$v1
q2 <- -pi/3; f2 <- 0.15;
b$v2 <- sin(f2*2*pi*(b$x*cos(q2)+b$y*sin(q2))+runif(1))
a2 <- array(0,c(dx,dy))
a2[as.matrix(b[,1:2])] <- b$v2
a <- array(0,c(dx,dy))
a[as.matrix(b[,1:2])] <- b$v1 + 2.5*b$v2
noise <- matrix(rnorm(dx*dy,0,1),ncol=dy)

persp(1:(dx/2), 1:(dy/2), a1[1:(dx/2), 1:(dy/2)], zlab="",
main="wave #1", theta=0, phi=45, ticktype="detailed", col="lightblue")
persp(1:(dx/2), 1:(dy/2), a2[1:(dx/2), 1:(dy/2)],
main="wave #2", theta=90, phi=-110, ticktype="detailed", col="lightblue")
persp(1:(dx/2), 1:(dy/2), a[1:(dx/2), 1:(dy/2)],
main="wave #1 + #2 ", theta=90, phi=-110, ticktype="detailed", col="lightblue")
persp(1:(dx/2), 1:(dy/2), a[1:(dx/2), 1:(dy/2)] + 5*noise[1:(dx/2), 1:(dy/2)],
main="wave #1 + #2 + 5*noise", theta=90, phi=-110, ticktype="detailed", col="lightblue")

image(x=1:dim(a1)[1] , y=1:dim(a1)[2], z=a1)
box(); mtext("wave #1")
image(x=1:dim(a2)[1] , y=1:dim(a2)[2], z=a2)
box(); mtext("wave #2")
image(x=1:dim(a)[1] , y=1:dim(a)[2], z=a+0*noise)
box(); mtext("wave #1 + #2 ")
image(x=1:dim(a)[1] , y=1:dim(a)[2], z=a+7*noise)
box(); mtext("wave #1 + #2 + 7*noise")

rc0 <- kz.rc2(a+0*noise, angle=c(q1,q2)*180/pi,f=c(f1,f2), m = 50, avg=FALSE)
cor(as.vector(a[1:dim(rc0)[1],1:dim(rc0)[2]]), as.vector(rc0), use="pairwise.complete.obs")

rc0 <- kz.rc2(a+0*noise, angle=c(q1,q2)*180/pi,f=c(f1,f2), m = 50, avg=TRUE, rlvl=15)
cor(as.vector(a[1:dim(rc0)[1],1:dim(rc0)[2]]), as.vector(rc0), use="pairwise.complete.obs")

rc <- kz.rc2(a+7*noise, angle=c(q1,q2)*180/pi,f=c(f1,f2), m = 50, avg=TRUE, rlvl=15, plot=TRUE)
cor(as.vector(a[1:dim(rc)[1],1:dim(rc)[2]]), as.vector(rc), use="pairwise.complete.obs")
dev.new();image(x=1:dim(rc)[1] , y=1:dim(rc)[2], z=a[1:dim(rc)[1],1:dim(rc)[2]])
box();title("Signal without noise")

rc <- kz.rc2(a+7*noise, angle=q2*180/pi, f=f2, m = 50, avg=TRUE, rlvl=21, plot=TRUE)
cor(as.vector(a2[1:dim(rc)[1],1:dim(rc)[2]]), as.vector(rc), use="pairwise.complete.obs")
```

```
dev.new();image(x=1:dim(rc)[1] , y=1:dim(rc)[2], z=a2[1:dim(rc)[1],1:dim(rc)[2]])
box();title("Signal without noise")
```

---

optDR *Improve Accuracy of KZ Periodogram Estimation with Optimization*

Description

Functions in this group are designed to improve the estimated wave parameters based on optimization of KZ directional periodograms and 2D periodograms.

Usage

```
optDR(a, rec, delta = 0.005, ...)

optD2P(a, rec, ...)
```

Arguments

a	Data array. Wave signals plus noise.
rec	Data list. For optDR, it is the outputs of function kzpdr. It includes the marked spectrum spike frequency values, directions.
delta	Searching range parameter for optimization. Default is 0.005.
...	Other arguments.

- k : Integer. The iterations number of KZFT.
- n : The sampling frequency rate as a multiplication of the Fourier frequencies

Details

optDR optimizes estimations of directional periodograms using R function stats::optimize. optD2P works for estimations of 2D periodograms; related optimization process is based on function stats::optim.

Value

optDR will return the data frame of detailed estimation for each direction.

See Also

kzpdr, kzp2

Examples

```
dx <- 300 # x range
dy <- 300 # y range
b <- expand.grid(x=1:dx, y=1:dy)
q1 <- pi/3; f1 <- 0.2;
b$v1 <- sin(f1*2*pi*(b$x*cos(q1)+b$y*sin(q1))+100*runif(1))
q2 <- pi/6; f2 <- 0.05;
b$v2 <- sin(f2*2*pi*(b$x*cos(q2)+b$y*sin(q2))+100*runif(1))
a <- array(0,c(dx,dy))
a[as.matrix(b[,1:2])] <- b$v1 + 1.5*b$v2
noise <- 5*matrix(rnorm(dx*dy,0,1),ncol=dy)

# Identifying with 2D periodogram
# kzp2.demo <- kzp2(a+noise)$kzp2d
QF <- kzp2.summary(kzp2.demo, num=2)

# Optimization of the 2D periodogram
# It may take 1 to 5 minutes
# kzp2.QF <- optD2P(a+noise, QF, k1=1, n1=1)
kzp2.QF

# Optimization of directional periodogram
# It may take 10 to 20 minutes
# kzpdr.demo <- kzpdr(a+noise, c(0,45,15,35)*pi/180, plot=TRUE, dpct=0.05)$rec
# kzpde.QF <- optDR(a+noise, kzpdr.demo)

QF2 <- kzpdr.eval(kzpdr.demo)
opt.QF2 <- kzpdr.eval(kzpdr.QF)
```

`smpg` *Smooth and Plot One Dimensional Kolmogorov-Zurbenko Periodogram*

Description

`kz.smpg` is designed to smooth and plot 1D KZ periodogram easily. It will calculate the raw periodogram, mark the spikes, smooth the periodogram, and then output the plot.

Usage

```
kz.smpg(x, dpct = 0.01, rg = c(0, 0.5), log = F, plot = F, ...)
```

Arguments

`x`	The data vector for analyses. Missing values are allowed.
`dpct`	A pre-specified percentage of total variation. Defaults to 1%.
`rg`	The frequency range of the outputted periodogram. Default is 0 to 0.5.

`log` TRUE or FLASE. Use log scale for output periodogram. Defaults to FLASE.

`plot` TRUE or FLASE. Flag for output periodogram plot or not. Defaults to FLASE.

`...` Other arguments.

- `m :` The window size for a regular Fourier transform
- `k :` The number of iterations for the KZFT
- `n :` The sampling frequency rate as a multiplication of the Fourier frequencies
- `p :` The distance between two successive intervals as a percentage of the total length of the data series
- `w :` Size of smoothing window. Default value is 20.
- `lvl :` "min" or "max". Threshold strategy for marking frequency spikes. "min" is used for cases with weak singles mixed with dominating strong spikes. Defaults to "max".
- `cut :` Set the minimum value for a marked frequency spike. Recommend to use argument `lvl` instead of setting this value directly.

Details

The smoothing process is based on a modified DiRienzo-Zurbenko (DZ) method, for which the smoothing window is not symmetric around the value point. The smoothing algorithm is implemented in C.

Value

Data frame for outputted periodogram, including column *spg* for the periodogram values, and *freq* for the frequencies.

See Also

`kzp`, `kzp2`, `kz.ft`

Examples

```
## Adapted from kzft::kzp example 2
t <- 1:2000
y <- 1.1*sin(2*pi*0.0339*t)+7*sin(2*pi*0.0366*t)+5*rnorm(length(t),0,1)
y[sample(t,100,replace=FALSE)] <- NA

## Not run:
# system.time(op <- kz.smpg(y, dpct=0.0001, rg=c(0.025,0.05),
# plot=TRUE, log=TRUE, lvl="min", n=10, k=2))

## End(Not run)
op <- kz.smpg(y, dpct=0.0000, f=c(0.0339,0.0366), rg=c(0.025,0.05),
n=10, k=2, plot=TRUE, lvl="min", log=FALSE)
```

Index