Package 'TDD'

February 19, 2015

Type Package

Title Time-Domain Deconvolution of Seismometer Response

Version 0.4

Date 2013-03-03

Author Jake Anderson

Maintainer Jake Anderson <ajakef@gmail.com>

Description

Deconvolution of instrument responses from seismic traces and seismogram lists from RSEIS. Includes pre-calculated instrument responses for several common instruments.

Depends R (>= 2.10), signal, RSEIS, pracma

License GPL

LazyLoad yes

Repository CRAN

Date/Publication 2013-10-04 07:48:29

NeedsCompilation no

R topics documented:

`ButterPZ` *Calculate Butterworth Filter Poles and Zeros*

Description

Calculates continuous poles and zeros (of the Laplace transform) of Butterworth filters.

Usage

```
ButterPZ(fl = NaN, nl = NaN, fh = NaN, nh = NaN, g = 1)
```

Arguments

`fl`	Low corner frequency (Hz)
`nl`	Order of high-pass filter
`fh`	High corner frequency (Hz)
`nh`	Order of low-pass filter
`g`	gain (unitless)

Details

For a bandpass filter, all inputs should be non-NaN. For high pass, fh and nh should be NaN; for low pass, fl and nl should be NaN. Input corner frequencies are in cycles/second, not radians/second.

Value

List including the following elements:

`poles`	Vector of poles (rad/s)
`zeros`	Vector of zeros (rad/s)
`np`	Number of poles
`nz`	Number of zeros
`Knorm`	Normalization constant
`Sense`	Sensitivity (V * s/m)

Author(s)

Jake Anderson

Examples

```
# Calculate poles and zeros of Butterworth filter with a second-order
# high-pass above 1 Hz, and a fourth-order low-pass below 10 Hz
ButterPZ(1, 2, 10, 4)
```

CalcCorners *Calculate Corner Frequencies*

Description

Inputs a continuous instrument response list and returns a vector of its cutoff frequencies (defined here as the -3dB point). Optionally plots results.

Usage

```
CalcCorners(PZ, f = 1:1000000/1000, PLOT = FALSE)
```

Arguments

PZ	Continuous instrument response list (from GetPZ, for example)
f	Vector of frequency values to test (Hz)
PLOT	Logical: plot response spectrum and mark corner frequencies?

Value

Vector of corner frequencies in response spectrum.

Author(s)

Jake Anderson

Examples

```
# Response of CMG-40T
PZ = GetPZ(12)[[1]]
CalcCorners(PZ, PLOT = TRUE)
```

COLOC *Seismic Data From Co-Located Seismometers*

Description

Example of seismic data structure. Channels 1-3 are from a broadband Guralp CMG-3T; channels 4-6 are from a short-period Guralp CMG-40T-1. Data were logged using a RefTek RT130 six-channel logger and are in volts. Deconvolution of instrument responses will demonstrate that the two sensors experience the same ground motion (see DeconSeis).

Usage

```
data(COLOC)
```

Format

List, consisting of:

JSTR list of digital seismic data traces

STNS vector of stations

dir directory

ifile original file names

COMPS Component names, V N E, e.g.

OCOMPS Old Component names

dt vector of delta-t, sampling time intervals

KNOTES Notes for plotting on panels

info List, detailed information about traces, including

dat not used

nn Number of traces

ex time axis for plotting

pcol colors for plotting

ok which traces are okay

wintim window span time, seconds

ftime alphanumeric time stamp

pickfile pickfile, see below

velfile velocity model list

stafile station information list including lat, lon, z

aname source name for loading

UWFILEID event ID number

The info list consists of:

fn file name

name identification name

yr start year

jd start julianday

mo month

dom day of month

hr hour

mi minute

sec second

msec millisecond

dt delta-t

t1 time 1

t2 time 2

off offset

n1 number of samples

n2 not used

n3 not used

n number of samples

See Also

DeconSeis

Examples

```
library(RSEIS)
data(COLOC)
swig(COLOC)
```

`ConvDiffCoef` *Convert Differential to Difference Eq. Coefficients*

Description

Returns difference equation coefficients corresponding to input differential equation coefficients and sample interval.

Usage

```
ConvDiffCoef(db, dt)
```

Arguments

db	Vector of differential equation coefficients
dt	Time interval (s)

Details

Input differential equation is of the form db[1] * f(t) + db[2] * f'(t) + db[3] * f"(t)...

Output difference equation is of the form b[1] * x_i + b[2] * x_(i-1) + b[3] * x_(i-2)...

Value

Coefficients of difference equation.

Author(s)

Jake Anderson

Examples

```
db = c(0, 0, 1) # represents f''(t)
dt = 0.1

ConvDiffCoef(db, dt)
```

ConvolveTrace *Convolve Trace with Instrument Response*

Description

Convolves a single velocity trace (m/s) with a discrete instrument response to get the voltage signal it returns.

Usage

```
ConvolveTrace(x, DPZ, dec = 1)
```

Arguments

x	Velocity trace (m/s)
DPZ	Discrete instrument response (from MakeDPZ, for example)
dec	Oversampling/decimation factor (optional)

Details

Discrete instrument responses are specific to a given sampling rate. If the response you give has a different sample rate (given by DPZ$dt) from the trace x, you will get incorrect results. DPZ$dt times dec should be equal to the sample interval of the trace.

Value

Convolved trace in volts (vector).

Author(s)

Jake Anderson

See Also

DeconTrace, DeconSeis

Examples

```
# Response of Guralp CMG-40T
DPZ = GetDPZ(12, 1)[[1]]

x = rnorm(1000)
ConvolveTrace(x, DPZ)
```

DeconSeis *Deconvolve discrete instrument response from many traces*

Description

Deconvolves instrument responses from a seismogram structure from the RSEIS package.

Usage

```
DeconSeis(GH, inst, L, fl = 0.1, fh = NaN, bitweight = NULL, dec =
rep(1, length(GH$JSTR)))
```

Arguments

GH	Seismogram structure
inst	Vector of indices of instrument responses within L to deconvolve from each trace in GH
L	List in which each element is a discrete instrument response (from MakeDPZ, for example)
fl	Low corner of filter (NaN for no high-pass filtering) (Hz)
fh	High corner of filter (NaN for no low-pass filtering) (Hz)
bitweight	Vector of optional counts-to-volts factors for data in counts (volts/counts)–NULL if data are already in volts
dec	Oversampling/decimation factor (optional); vector equal to the number of traces considered

Details

Discrete instrument responses are specific to a given sampling rate. If the response you give has a different sample rate (given by DPZ$dt) from the trace x, you will get incorrect results. DPZ$dt * dec should be equal to the sample intervals of the traces.

Value

GH, with the instrument response removed from every trace.

Author(s)

Jake Anderson

Examples

```
data(COLOC)
swig(COLOC)
L = GetDPZ(c(4, 14), c(0.01, 0.01)) # get responses for 3T and 40T-1s
inst = c(1,1,1,2,2,2) # deconvolve 3T response from channel 1-3,
                      # 40T-1 response from channel 4-6
D = DeconSeis(COLOC, inst, L)
swig(D)
```

DeconTrace *Deconvolve Instrument Response (Single Trace)*

Description

Deconvolves a discrete instrument response from a seismic trace.

Usage

```
DeconTrace(x, DPZ, fl = 0.05, fh = NaN, bitweight = NULL, dec = 1)
```

Arguments

x	Trace from which instrument response is deconvolved
DPZ	Discrete instrument response list (from MakeDPZ, for example)
fl	Low corner of filter (NaN for no high-pass filtering) (Hz)
fh	High corner of filter (NaN for no low-pass filtering) (Hz)
bitweight	Optional counts-to-volts factor for data in counts (volts/counts)–NULL if data are already in volts
dec	Oversampling/decimation factor (optional)

Details

Discrete instrument responses are specific to a given sampling rate. If the response you give has a different sample rate (given by DPZ$dt) from the trace x, you will get incorrect results. DPZ$dt * dec should be equal to the trace's sample interval.

Value

Deconvolved velocity trace (vector).

Author(s)

Jake Anderson

See Also

ConvolveTrace, DeconSeis

Examples

```
# Response of Guralp CMG-3T
DPZ = GetDPZ(4, 0.01)[[1]]

data(COLOC)
x = COLOC$JSTR[[1]]

DeconTrace(x, DPZ)
```

DPZLIST *List of Pre-Calculated Discrete Instrument Responses*

Description

List of discrete instrument responses of 14 common seismometers for 6 common sample intervals.

Usage

```
data(DPZLIST)
```

Format

List of 14 lists (each corresponding to a different seismometer), each including 6 lists (each corresponding to a different sample interval), each containing the following elements:

Sense Instrument passband sensitivity (V * s/m)

Knorm Normalization constant

poles Poles of Laplace transform of instrument response (rad/s)

np Number of poles

zeros Zeros of Laplace transform of instrument response (rad/s)

nz Number of zeros

dt Sample interval for this response (s)

fmax Maximum frequency for which this digital response matches the true analog response of the sensor within 1%

Zpg Zpg-class element (from package 'signal') giving the digital response of the filter in terms of its zeros and poles (in Z-transform space) and gain.

Details

Seismometers are numbered as follows:

Broadband Seismometers: 1. Streckeisen STS-1 (360 s) 2. Trillium 240 (generation 1) 3. Trillium 240 (generation 2) 4. Guralp CMG-3T 5. Streckeisen STS-2 (generation 1) 6. Streckeisen STS-2 (generation 2) 7. Streckeisen STS-2 (generation 3) 8. Trillium 120 9. Compact Trillium

Intermediate Seismometers: 10. Trillium 40 11. Guralp CMG-3ESP 12. Guralp CMG-40T (30 s) 13. Streckeisen STS-1 (20 s)

Short-period Seismometers: 14: Guralp CMG-40T (1 s)

Digital responses are provided for the following six sample intervals (in seconds): 1, 0.1, 0.05, 0.025, 0.02, 0.01.

Note

The STS-2 and Trillium 240 come in multiple generations, each with a slightly different response. For the Trillium 240, serial numbers less than 400 belong to generation 1, while serial numbers greater than or equal to 400 are in generation 2. To determine which generation an STS-2 is, see http://www.iris.edu/NRL/sensors/streckeisen/streckeisen_sts2_sensors.htm .

Many short-period instruments were intentionally omitted because their responses depend on installation-specific parameters and are therefore not completely standardized. Given that discrete responses of short-period instruments can be calculated quickly, I consider the convenience of having pre-calculated responses for these instruments to not be worth the risk of the user selecting the wrong response and getting inaccurate results.

Examples

```
# 40T, 0.01-s response:
data(DPZLIST)
DPZ = DPZLIST[[12]][[6]]
```

GetDPZ *Retrieve Pre-Calculated Discrete Instrument Response*

Description

Discrete instrument responses for common seismometers and sample rates have been pre-calculated and included in this package. GetDPZ retrieves them.

Usage

```
GetDPZ(w, dt)
```

Arguments

w	Vector of indices of seismometers used (see Details)
dt	Sample intervals corresponding to instruments in w

Details

Seismometers are numbered as follows:

Broadband Seismometers: 1. Streckeisen STS-1 (360 s) 2. Trillium 240 (generation 1) 3. Trillium 240 (generation 2) 4. Guralp CMG-3T 5. Streckeisen STS-2 (generation 1) 6. Streckeisen STS-2 (generation 2) 7. Streckeisen STS-2 (generation 3) 8. Trillium 120 9. Compact Trillium

Intermediate Seismometers: 10. Trillium 40 11. Guralp CMG-3ESP 12. Guralp CMG-40T (30 s) 13. Streckeisen STS-1 (20 s)

Short-period Seismometers: 14: Guralp CMG-40T (1 s)

Value

List of instrument responses corresponding to instruments and sample intervals given in w and dt.

Note

The STS-2 and Trillium 240 come in multiple generations, each with a slightly different response. For the Trillium 240, serial numbers less than 400 belong to generation 1, while serial numbers greater than or equal to 400 are in generation 2. To determine which generation an STS-2 is, see http://www.iris.edu/NRL/sensors/streckeisen/streckeisen_sts2_sensors.htm .

Many short-period instruments were intentionally omitted because their responses depend on installation-specific parameters and are therefore not completely standardized. Given that discrete responses of short-period instruments can be calculated quickly, I consider the convenience of having pre-calculated responses for these instruments to not be worth the risk of the user selecting the wrong response and getting inaccurate results.

Author(s)

Jake Anderson

References

Sources for all instrument responses are given in the comments of GetPZ; the reference list is too long to include here.

See Also

GetPZ MakeDPZ

Examples

```
# responses for 3T sampling at 1 Hz and 40T (30 s) sampling at 40 Hz
DPZLIST = GetDPZ(c(4, 12), c(1, 0.025))
```

GetPZ *Retrieve Included Continuous Instrument Response*

Description

Continuous responses for common seismometers are included in this package. GetPZ retrieves them.

Usage

```
GetPZ(w)
```

Arguments

w	Vector of indices of seismometers used (see Details)

Details

Seismometers are numbered as follows:

Broadband Seismometers: 1. Streckeisen STS-1 (360 s) 2. Trillium 240 (generation 1) 3. Trillium 240 (generation 2) 4. Guralp CMG-3T 5. Streckeisen STS-2 (generation 1) 6. Streckeisen STS-2 (generation 2) 7. Streckeisen STS-2 (generation 3) 8. Trillium 120 9. Compact Trillium

Intermediate Seismometers: 10. Trillium 40 11. Guralp CMG-3ESP 12. Guralp CMG-40T (30 s) 13. Streckeisen STS-1 (20 s)

Short-period Seismometers: 14: Guralp CMG-40T (1 s)

Value

List of instrument responses corresponding to instruments given in w.

Note

The STS-2 and Trillium 240 come in multiple generations, each with a slightly different response. For the Trillium 240, serial numbers less than 400 belong to generation 1, while serial numbers greater than or equal to 400 are in generation 2. To determine which generation an STS-2 is, see http://www.iris.edu/NRL/sensors/streckeisen/streckeisen_sts2_sensors.htm .

Certain short-period instruments were intentionally omitted because their responses depend on installation-specific parameters and are therefore not completely standardized. Given that discrete responses of short-period instruments can be calculated quickly, I consider the convenience of having pre-calculated responses for these instruments to not be worth the risk of the user selecting the wrong response and getting inaccurate results.

Author(s)

Jake Anderson

References

Sources for all instrument responses are given in the comments of GetPZ; the reference list is too long to include here.

See Also

ReadInstr MakeRespSP GetDPZ MakeDPZ

Examples

```
# responses for 3T sampling at 1 Hz and 40T (30 s)
PZLIST = GetPZ(c(4, 12))
```

MakeDPZ *Calculate Find Digital Match to Analog Poles/Zeros*

Description

Calculates digital poles and zeros to match a continuous instrument response given in poles, zeros, and sensitivity. Discretization effects mean that the analog poles and zeros do not work for finite sample rates, with discrepancies increasing as the time interval increases.

This function uses two methods to match the responses. The first uses a finite difference approximation and Markov Chain Monte Carlo (MCMC) routine to optimize the response. The second uses the bilinear transform to approximate the analog response. Whichever of these responses best matches the analog response is used; usually, the first method provides a better fit.

Usage

```
MakeDPZ(PZ, dt, fmin = 1/360, niter = 50000, ...)
```

Arguments

PZ	List including poles and zeros of instrument response
dt	Sample interval (s)
fmin	Lowest frequency to match (Hz)
niter	Number of iterations in Markov Chain Monte Carlo
...	Additional arguments for MatchCoefDPZ

Details

Large N allow it to match very low frequencies, but take longer to calculate. Large niter means longer calculation time, but probably a closer match. The burn-in period should be set to zero unless you want the posterior distribution of the poles and zeros. Large sigfac means that the MCMC makes smaller jumps, meaning it explores the sample space more slowly, but is less likely to make large jumps away from the interesting region. Note that the standard deviations of proposal distributions of the model parameters are proportional to the magnitude of the "guess" model– meaning that model parameters identically equal to zero (such as zeros at the origin) are fixed.

Discretization effects often make it difficult to match higher frequencies. Close match of somewhat high frequencies is done at the expense of poor match of very high frequencies. If very high frequencies are not interesting, fh should be left at its default value. Otherwise, it should be set to the highest interesting frequency.

Value

List including the following elements:

poles	Vector of "analog poles" (rad/s)
zeros	Vector of "analog zeros" (rad/s)
np	Number of poles
nz	Number of zeros
Knorm	Normalization constant
Sense	Sensitivity (V * s/m)
dt	Sample interval (s)
fmax	Maximum frequency for which this digital response matches the true analog response of the sensor within 1%
Zpg	Zpg-class element (from package 'signal') giving the digital response of the filter in terms of its zeros and poles (in Z-transform space) and gain.

Note

This is a wrapper function for MatchCoefDPZ, and should be used for most applications.

Author(s)

Jake Anderson

See Also

MatchCoefDPZ

Examples

```
# Response of Guralp CMG-40T

PZ = list(poles = c(-0.149 + 0.149i, -0.149 - 0.149i, -503, -1010,
-1130), zeros = c(0, 0), Knorm = 574095649, Sense = 800)
# MakeDPZ(PZ, dt = 0.01, fmin = 1/60) # takes minutes to run
```

`MakeRespSP` *Calculate Poles and Zeros of Short-Period Sensor*

Description

Many mechanical short-period seismometers are characterized by three parameters: the natural angular frequency (omega naught), damping coefficient (h), and sensitivity. The response of these sensors to a velocity impulse has two poles and two zeros; the response is zero at the origin, increases roughly proportionately with f^2 up to a low corner, and is flat above the low corner.

The differential equation describing this, where y is the output voltage and v is the velocity of the ground, is

y" + 2*h*omega_0*y' + omega_0^2*y = Sense * v" Some short-period sensors are customizable, so it is very important to make sure you use the correct parameters for your installation here.

Usage

```
MakeRespSP(h, omega0, Sense, f_Sense = NULL)
```

Arguments

`h`	Damping coefficient (unitless, 1 for critical damping)
`omega0`	Natural angular frequency (rad/s)
`Sense`	Sensitivity in passband (V * s/m)
`f_Sense`	If given, the frequency (Hz) at which Sense is valid. If NULL (which should ordinarily be the case), Sense is assumed to be valid at frequencies much higher than the low corner.

Value

List including the following elements:

`Sense`	Sensitivity of instrument (V * s/m)
`Knorm`	Normalization constant
`poles`	Poles of Laplace transform of instrument impulse response (rad/s)

np	Number of poles
zeros	Zeros of Laplace transform of instrument impulse response (rad/s)
nz	Number of zeros

Author(s)

Jake Anderson

See Also

GetPZ ReadInstr

Examples

```
# L4C3D
omega0 = 2*pi # 1 Hz natural frequency * 2pi
h = 0.707
Sense = 171
MakeRespSP(h, omega0, Sense)
```

MatchCoefDPZ *Find Digital Match to Analog Poles/Zeros*

Description

Calculates digital poles and zeros to match a continuous instrument response given in poles, zeros, and sensitivity. Discretization effects mean that the given poles and zeros do not work for finite sample rates, with discrepancies increasing as the time interval increases. This function uses a Markov Chain Monte Carlo (MCMC) routine to match the responses.

Usage

```
MatchCoefDPZ(PZ, dt, N, niter = 50000, burn = 0, sigfac = 1, fh =
0.25/dt, k = 0.001, verbose = TRUE)
```

Arguments

PZ	List including poles and zeros of instrument response
dt	Sample interval (s)
N	Number of samples to use when matching response (higher to match lower frequencies)
niter	Number of iterations in Markov Chain Monte Carlo
burn	Burn-in period of MCMC
sigfac	Factor by which standard deviations are reduced in MCMC
fh	Highest frequency to try to match (default 0.25 * sampling rate)
k	Weight to give to misfit for frequencies over fh–should be low to prevent high frequencies from being matched at the expense of low frequencies
verbose	Logical: if TRUE, progress updates are printed to the screen

Details

Large N allow it to match very low frequencies, but take longer to calculate. Large niter means longer calculation time, but probably a closer match. The burn-in period should be set to zero unless you want the posterior distribution of the poles and zeros. Large sigfac means that the MCMC makes smaller jumps, meaning it explores the sample space more slowly, but is less likely to make large jumps away from the interesting region. Note that the standard deviations of proposal distributions of the model parameters are proportional to the magnitude of the "guess" model– meaning that model parameters identically equal to zero (such as zeros at the origin) are fixed.

Discretization effects often make it difficult to match higher frequencies. Close match of somewhat high frequencies is done at the expense of poor match of very high frequencies. If very high frequencies are not interesting, fh should be left at its default value. Otherwise, it should be set to the highest interesting frequency.

Value

List including the following elements:

`b`	Moving Average polynomial coefficients
`a`	Autoregressive polynomial coefficients
`analogresp`	Continuous "analog" response
`digitalresp`	Response of digital filter
`inv`	Detailed MCMC results
`error`	Geometric root-mean-square error between digital and analog response
`DPZ`	Digital Poles and Zeros

Note

MakeDPZ is a higher-level routine and should be used for most applications.

Author(s)

Jake Anderson

See Also

MakeDPZ

Examples

```
# Response of Guralp CMG-40T

PZ = list(poles = c(-0.149 + 0.149i, -0.149 - 0.149i, -503, -1010,
-1130), zeros = c(0, 0), Knorm = 574095649, Sense = 800)
# MatchCoefDPZ(PZ, dt = 0.01, N = 10000) # takes minutes to run
```

`Metropolis` *Metropolis-Hastings Markov Chain Monte Carlo*

Description

Uses the Metropolis-Hastings Markov Chain Monte Carlo (MCMC) method to determine an optimal model to fit some data set.

Usage

```
Metropolis(loglikelihood, sigma, m1, niter, gen, logproposal, logprior =
function(x) 0, burn = 0, save_int = 10, verbose, ...)
```

Arguments

`loglikelihood`	Function to calculate the log of a model's likelihood
`sigma`	Vector of standard deviations to use when generating a new model
`m1`	Starting "guess" model
`niter`	Number of iterations to run
`gen`	Function to generate a new model
`logproposal`	Function to calculate the proposal distribution for a new model
`logprior`	Function to calculate the log of the prior distribution value of a model
`burn`	Initial "burn-in" period from which results are not saved
`save_int`	Number of iterations between saved models
`verbose`	Logical: if TRUE, progress updates are printed to the screen
`...`	Parameters to pass to loglikelihood

Details

Metropolis prints progress information to the screen every 1000 iterations. These lines include the following:

Number of iterations completed out of total loglikelihood of current model loglikelihood of proposed model loglikelihood of best model found so far Whether the proposed model this round is rejected or accepted Acceptance ratio over the last 100 iterations

Value

List including the following elements:

`m`	Matrix where each row is the test model parameters of an iteration
`l`	log-likelihood of each iteration's model
`a`	Acceptance ratio (moving window of length 100)
`best`	List including best model found and its log-likelihood

Author(s)

Jake Anderson

References

Hastings, W.K. (1970). "Monte Carlo Sampling Methods Using Markov Chains and Their Applications". Biometrika 57 (1): 97-109.

Aster, R.C., Borchers, B., Thurber, C.H., Parameter Estimation and Inverse Problems, Elsevier Academic Press, 2012.

Examples

```
# try to find a non-negative vector m so that
# 1.  sqrt(m[1]^2 + m[2]^2) is close to 5
# 2.  sqrt(m[1] * m[2]) is close to 3.5
# 3.  2 * m[1] + m[2] is close to 10

# We are trying to match this data vector:
data = c(5, 3.5, 10)

# Define log-likelihood as -0.5 * sum of squared differences between
# modeled and true data
loglikelihood = function(model, data){
  d2 = c(sqrt(sum(model^2)), sqrt(abs(prod(model))), sum(model*c(2,1)))
  -0.5 * sum((d2 - data)^2)
}

# A proposed model is generated by randomly picking a model parameter
# and perturbing it by a random number distributed normally according to sigma
generate = function(x, sigma){
  w = ceiling(runif(1) * length(x))
  x[w] = x[w] + rnorm(1, 0, sigma[w])
  return(x)
}

# Proposal distribution is defined as multivariate normal, with mean
# zero and standard deviations sigma:
logproposal = function(x1, x2, sigma){
  -0.5 * sum(((x1) - (x2))^2/(sigma+1e-12)^2)
}

# logprior reflects prior knowledge that the answer is non-negative
logprior = function(m){
  if(all(m >= 0))
    0
  else
    -Inf
}


sigma = c(0.1, 0.1)
```

```
m1 = c(0, 0)
x = Metropolis(loglikelihood, sigma, m1, niter = 5000, gen = generate,
logproposal = logproposal, logprior = logprior, burn = 0, save_int = 1,
data = data)

# Notice the high acceptance ratios--this means that values in sigma are
# too low.  The MCMC is probably "optimally tuned" when sigma is set so
# acceptance ratios vary between roughly 0.2 and 0.5.

# Plot models--par 1 on x, par 2 on y axis
# Note initial trajectory away from m1 (0, 0) to more likely
# region--this can be eliminated by setting 'burn' to a higher value
plot(x$m[,1], x$m[,2], pch = '.', col = rainbow(nrow(x$m)))

# Histograms/scatter plots showing posterior distributions.
# Note the strong covariance between these parameters!
par(mfrow = c(2, 2))
hist(x$m[,1])
plot(x$m[,2], x$m[,1], pch = '.')
plot(x$m[,1], x$m[,2], pch = '.')
hist(x$m[,2])
```

`Oversample` *Oversample by Nearest-Neighbor Interpolation*

Description

In order to maintain digital filter fidelity at very high frequencies, it is sometimes necessary to oversample a signal. This function oversamples a signal by nearest-neighbor interpolation.

Usage

```
Oversample(x, n)
```

Arguments

`x`	Signal to be oversampled
`n`	Factor by which it should be oversampled

Details

The output probably needs to be decimated after deconvolution.

Value

Vector of oversampled data.

Author(s)

Jake Anderson

Examples

```
# Oversample a random trace by a factor of 10
x = rnorm(100)
Oversample(x, 10)
```

PlotResp *Plot Instrument Responses*

Description

Plots responses of analog and digital poles/zeros lists.

Usage

```
PlotResp(PZ, DPZ, fmin = 0.01)
```

Arguments

PZ	Poles and zeros of continuous instrument response
DPZ	Digital poles and zeros of discrete instrument response
fmin	Minimum frequency to plot

Details

PZ and DPZ must contain the elements poles, zeros, np, nz, Knorm, and Sense. Additionally, DPZ must contain the element dt.

Value

Graphical side effects only.

Author(s)

Jake Anderson

See Also

PZ2Resp

Examples

```
# Response of Guralp CMG-40T

PZ = GetPZ(12)[[1]]
DPZ = GetDPZ(12, 1)[[1]]
PlotResp(PZ, DPZ, fmin = 1/50)
```

PZ2Coef *Calculate Recursive Filter Coefficients*

Description

Returns coefficients of recursive filter approximating instrument response, given poles/zeros and sample interval.

Usage

```
PZ2Coef(PZ, dt)
```

Arguments

PZ	Poles/zeros list
dt	Sample interval

Details

PZ requires the following elements: poles: Vector of poles np: number of poles zeros: Vector of zeros nz: Number of zeros Knorm: Normalization constant Sense: Instrument sensitivity (V/(m/s))

Output recursive filter is of the form a[1] * y_i + a[2] * y_(i-1) + a[3] * y_(i-2) + ... = b[1] * x_i + b[2] * x_(i-1) + b[3] * x_(i-2) + ..., where x is ground motion velocity and y is the recorded voltage.

Value

List including the following elements:

b	Coefficients of filter input terms
a	Coefficients of filter output terms

Note

Due to effects of discretization, the spectrum of the recursive filter DOES NOT match that of the poles/zeros. So, poles and zeroes must be adjusted in order to make them match, either by inversion or by the bilinear transform.

Author(s)

Jake Anderson

Examples

```
PZ_40T = list(poles = c(-0.149 + 0.149i, -0.149 - 0.149i, -503, -1010,
-1130), zeros = c(0, 0), Knorm = 574095649, Sense = 800)

dt = 0.01
PZ2Coef(PZ_40T, dt)
```

PZ2Resp *Calculate Instrument Response from Poles/Zeros*

Description

Returns complex instrument response for a vector of frequencies and set of analog poles and zeros. Optionally plots the magnitude of the instrument response.

Usage

```
PZ2Resp(PZ, f, PLOT = TRUE)
```

Arguments

PZ	Analog poles and zeros
f	Vector of frequencies for which response is calculated
PLOT	Logical: whether to plot magnitude of response

Details

The response is calculated by the following equation: R(s) = PZ$Sense * PZ$Knorm * prod(s - PZ$zeros)/prod(s - PZ$poles)

where s = 2 * pi * 1i * f.

PZ requires the following elements: poles: Vector of poles np: number of poles zeros: Vector of zeros nz: Number of zeros Knorm: Normalization constant Sense: Instrument sensitivity (V/(m/s))

Value

Vector of instrument response values corresponding to the frequencies in f.

Author(s)

Jake Anderson

Examples

```
# Response of Guralp CMG-40T

PZ = GetPZ(12)[[1]]

f = (1:10000 - 1)/1000

PZ2Resp(PZ, f)
```

PZLIST *List of Continuous Instrument Responses*

Description

List of continuous instrument responses of 14 common seismometers.

Usage

```
data(PZLIST)
```

Format

List of 14 lists, each consisting of:

Sense Instrument passband sensitivity (V * s/m)

Knorm Normalization constant

poles Poles of Laplace transform of instrument response (rad/s)

np Number of poles

zeros Zeros of Laplace transform of instrument response (rad/s)

nz Number of zeros

Details

Seismometers are numbered as follows:

Broadband Seismometers: 1. Streckeisen STS-1 (360 s) 2. Trillium 240 (generation 1) 3. Trillium 240 (generation 2) 4. Guralp CMG-3T 5. Streckeisen STS-2 (generation 1) 6. Streckeisen STS-2 (generation 2) 7. Streckeisen STS-2 (generation 3) 8. Trillium 120 9. Compact Trillium

Intermediate Seismometers: 10. Trillium 40 11. Guralp CMG-3ESP 12. Guralp CMG-40T (30 s) 13. Streckeisen STS-1 (20 s)

Short-period Seismometers: 14: Guralp CMG-40T (1 s)

Note

The STS-2 and Trillium 240 come in multiple generations, each with a slightly different response. For the Trillium 240, serial numbers less than 400 belong to generation 1, while serial numbers greater than or equal to 400 are in generation 2. To determine which generation an STS-2 is, see http://www.iris.edu/NRL/sensors/streckeisen/streckeisen_sts2_sensors.htm .

Certain short-period instruments were intentionally omitted because their responses depend on installation-specific parameters and are therefore not completely standardized. Given that discrete responses of short-period instruments can be calculated quickly, I consider the convenience of having pre-calculated responses for these instruments to not be worth the risk of the user selecting the wrong response and getting inaccurate results.

References

Each of these responses was drawn from either a manufacturer document or from IRIS; sources are noted in the comments of GetPZ.

Examples

```
# 40T response:
data(PZLIST)
PZ = PZLIST[[12]]
```

`ReadInstr` *Read IRIS Instrument Response File*

Description

Scans an instrument response file from IRIS and returns a list with poles, zeros, normalization constant, and sensitivity.

Usage

```
ReadInstr(fn)
```

Arguments

`fn`	List of filenames of instrument response files to read

Value

List including the following elements:

`Sense`	Sensitivity of instrument (V * s/m)
`Knorm`	Normalization constant
`poles`	Poles of Laplace transform of instrument impulse response (rad/s)
`np`	Number of poles
`zeros`	Zeros of Laplace transform of instrument impulse response (rad/s)
`nz`	Number of zeros

Author(s)

Jake Anderson

See Also

GetPZ MakeRespSP

Examples

```
# not run:
# ReadInstr('SAC_PZs_IU_OTAV_BHZ_00_2009.091.00.00.00.0000_2010.136.22.12.60.99999')
```

Index