Package 'PEIP'

August 21, 2023

Type Package

Title Geophysical Inverse Theory and Optimization

Version 2.2-5

Date 2023-08-09

Depends R (>= 2.12)

Imports bvls, Matrix, RSEIS, pracma, geigen, fields

Author Jonathan M. Lees [aut, cre]

Maintainer Jonathan M. Lees <jonathan.lees@unc.edu>

Description
Several functions introduced in Aster et al.'s book on inverse theory. The functions are often translations of MATLAB code developed by the authors to illustrate concepts of inverse theory as applied to geophysics. Generalized inversion, tomographic inversion algorithms (conjugate gradients, 'ART' and 'SIRT'), non-linear least squares, first and second order Tikhonov regularization, roughness constraints, and procedures for estimating smoothing parameters are included.

NeedsCompilation no

Repository CRAN

Date/Publication 2023-08-21 08:52:41 UTC

R topics documented:

`PEIP-package` *Inverse Theory Functions for PEIP book*

Description

Auxilliary functions and routines for running the examples and excersizes described in the book on inverse theory.

Details

These functions are used in conjunction with the example described in the PEIP book.

There is one C-code routine, interp2grid. This is introduced to replicate the MATLAB code interp2. It does not work exactly as the matlab code prescribes.

In the PEIP library one LAPACK routine is called: dggsvd. In R, LAPACK routines are stored in slightly different locations on Linux, Windows and Mac computers. Be aware. This will come up in examples from Chapter 4.

Almost all examples work as scripts run with virtually no user input, e.g.

Author(s)

Jonathan M. Lees<jonathan.lees.edu> Maintainer:Jonathan M. Lees<jonathan.lees.edu>

References

Aster, R.C., C.H. Thurber, and B. Borchers, *Parameter Estimation and Inverse Problems*, Elsevier Academic Press, Amsterdam, 2005.

`Ainv` *An Inverse Solution*

Description

QR decomposition solution to Ax=b

Usage

```
Ainv(GAB, x, tol = 1e-12)
```

Arguments

`GAB`	design matrix
`x`	right hand side
`tol`	tolerance for singularity

Details

I needed something to make up for the lame-o matlab code that does this h = G\x to get the inverse

Value

Inverse Solution

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

Examples

```
set.seed(2015)
GAB = matrix(runif(36), ncol=6)
truex =rnorm(ncol(GAB))
rhs = GAB %*% truex

rhs = as.vector(rhs )

tout = Ainv(GAB, rhs, tol = 1e-12)
tout - truex
```

---

art — *ART Inverse solution*

Description

ART algorythm for solving sparse linear inverse problems

Usage

```
art(A, b, tolx, maxiter)
```

Arguments

A	Constraint matrix
b	right hand side
tolx	difference tolerance for successive iterations (stopping criteria)
maxiter	maximum iterations (stopping criteria).

Details

Alpha is a damping factor. If alpha<1, then we won't take full steps in the ART direction. Using a smaller value of alpha (say alpha=.75) can help with convergence on some problems.

Value

x	solution

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

References

Aster, R.C., C.H. Thurber, and B. Borchers, *Parameter Estimation and Inverse Problems*, Elsevier Academic Press, Amsterdam, 2005.

Examples

```
set.seed(2015)
G = setDesignG()
### % Setup the true model.
mtruem=matrix(rep(0, 16*16), ncol=16,nrow=16);

mtruem[9,9]=1; mtruem[9,10]=1; mtruem[9,11]=1;
mtruem[10,9]=1; mtruem[10,11]=1;
mtruem[11,9]=1; mtruem[11,10]=1; mtruem[11,11]=1;
mtruem[2,3]=1; mtruem[2,4]=1;
mtruem[3,3]=1; mtruem[3,4]=1;

### % reshape the true model to be a vector
mtruev=as.vector(mtruem);

### % Compute the data.
dtrue=G %*% mtruev;

### % Add the noise.


d=dtrue+0.01*rnorm(length(dtrue));

mkac<-art(G,d,0.01,200)
par(mfrow=c(1,2))
imagesc(matrix(mtruem,16,16) , asp=1 , main="True Model" );

imagesc(matrix(mkac,16,16) , asp=1 , main="ART Solution" );
```

bartl *Bartlett window*

Description

Bartlett (triangle) window of length m

Usage

```
bartl(m)
```

Arguments

m	integer, length of vector

Value

vector

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

References

Aster, R.C., C.H. Thurber, and B. Borchers, *Parameter Estimation and Inverse Problems*, Elsevier Academic Press, Amsterdam, 2005.

Examples

```
bartl(11)
```

---

bayes *Bayes Inversion*

Description

Given a linear inverse problem Gm=d, a prior mean mprior and covariance matrix covm, data d, and data covariance matrix covd, this function computes the MAP solution and the corresponding covariance matrix.

Usage

```
bayes(G, mprior, covm, d, covd)
```

Arguments

`G`	Design Matrix
`mprior`	vector, prior model
`covm`	vector, model covariance
`d`	vector, right hand side
`covd`	vector, data covariance

Value

vector model

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

References

Aster, R.C., C.H. Thurber, and B. Borchers, *Parameter Estimation and Inverse Problems*, Elsevier Academic Press, Amsterdam, 2005.

Examples

```
## Not run:
set.seed(2015)
G = setDesignG()
###
mtruem=matrix(rep(0, 16*16), ncol=16,nrow=16);

mtruem[9,9]=1; mtruem[9,10]=1; mtruem[9,11]=1;
mtruem[10,9]=1; mtruem[10,11]=1;
mtruem[11,9]=1; mtruem[11,10]=1; mtruem[11,11]=1;
mtruem[2,3]=1; mtruem[2,4]=1;
mtruem[3,3]=1; mtruem[3,4]=1;

###
mtruev=as.vector(mtruem);
imagesc(matrix(mtruem,16,16) , asp=1 , main="True Model" );


matrix(mtruem,16,16) , asp=1 , main="True Model" )


###
dtrue=G %*% mtruev;

###
d=dtrue+0.01*rnorm(length(dtrue));
covd = 0.1*diag( nrow=length(d) )
covm = 1*diag( nrow=dim(G)[2] )

## End(Not run)
```

`blf2` *Bounded least squares*

Description

Bounded least squares

Usage

```
blf2(A, b, c, delta, l, u)
```

Arguments

A	Design Matrix
b	Right hand side
c	matrix weight on x
delta	tolerance
l	lower bound
u	upper bound

Details

Solves the problem: min/max c'*x where || Ax-b || <= delta and l <= x <= u.

Value

x	solution

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

References

Aster, R.C., C.H. Thurber, and B. Borchers, *Parameter Estimation and Inverse Problems*, Elsevier Academic Press, Amsterdam, 2005.

Stark, P.B. , and R. L. Parker, *Bounded-Variable Least-Squares: An Algorithm and Applications*, Computational Statistics 10:129-141, 1995.

Examples

```
###  set up an inverse problem:Shaw problem

n = 20
G = shawG(n,n)

spike = rep(0,n)
spike[10] = 1

spiken = G %*%  spike

wts = rep(1, n)
delta = 1e-03
set.seed(2015)
dspiken = spiken + 6e-6 *rnorm(length(spiken))

lb = spike - (.2) * wts
ub = spike + (.2) * wts

dspiken = dspiken
```

```
blf2(G, dspiken, wts , delta, lb, ub)
```

`cgls` *Conjugate gradient Least squares*

Description

Conjugate gradient Least squares

Usage

```
cgls(Gmat, dee, niter)
```

Arguments

`Gmat`	input matrix
`dee`	right hand side
`niter`	max number of iterations

Details

Performs niter iterations of the CGLS algorithm on the least squares problem min norm(G*m-d). Gmat should be a sparse matrix.

Value

`X`	matrix of models
`rho`	misfit norms
`eta`	model norms

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

References

Aster, R.C., C.H. Thurber, and B. Borchers, *Parameter Estimation and Inverse Problems*, Elsevier Academic Press, Amsterdam, 2005.

Examples

```
set.seed(11)
####  perfect data with no noise
n <- 5
A <- matrix(runif(n*n),nrow=n)
B <- runif(n)
###  get right-hand-side (data)
trhs = as.vector( A %*% B  )
Lout = cgls(A, trhs , 15)

###  solution is
Lout$X[,15]

Lout$X[,15] - B
```

---

chi — *Chi function*

Description

Chi function

Usage

```
chi(x, n)
```

Arguments

x	value
n	degrees of freedom

Value

function evaluated

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

References

Aster, R.C., C.H. Thurber, and B. Borchers, *Parameter Estimation and Inverse Problems*, Elsevier Academic Press, Amsterdam, 2005.

Examples

```
x = seq(0, 10, length=100)
n = 5
y=chi(x, n)
plot(x, y)
```

---

chi2cdf *Chi-Sq CDF*

Description

Computes the Chi^2 CDF, using a transformation to N(0,1) on page 333 of Thistead, Elements of Statistical Computing.

Usage

```
chi2cdf(x, n)
```

Arguments

x	end value of chi^2 pdf to integrate to. (scalar)
n	degrees of freedom (scalar)

Details

Note that x and m must be scalars.

Value

p	probability that Chi^2 random variable is less than or equal to x (scalar).

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

References

Aster, R.C., C.H. Thurber, and B. Borchers, *Parameter Estimation and Inverse Problems*, Elsevier Academic Press, Amsterdam, 2005.

Examples

```
x= seq(from=0.1, to=0.9, length=20)
chi2cdf(x , 3)
```

chi2inv *Inverse Chi-Sq*

Description

Inverse Chi-Sq

Usage

```
chi2inv(x, n)
```

Arguments

`x`	probability that Chi^2 random variable is less than or equal to x (scalar).
`n`	degrees of freedom(scalar)

Details

Computes the inverse Chi^2 distribution corresponding to a given probability that a Chi^2 random variable with the given degrees of freedom is less than or equal to x. Uses chi2cdf.m.

Value

corresponding value of x for given probability.

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

References

Aster, R.C., C.H. Thurber, and B. Borchers, *Parameter Estimation and Inverse Problems*, Elsevier Academic Press, Amsterdam, 2005.

See Also

chi, chi2cdf

Examples

```
x = seq(from=0.1, to=0.9, length=10)
h = chi2cdf(x, 3)

chi2inv(h, 3)
```

dcost *cosine transform*

Description

Computes the column-by-column discrete cosine transform of X.

Usage

```
dcost(X)
```

Arguments

X	Time series matrix

Value

cosine transformaed data

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

Examples

```
 x <- 1:4

### compare fft with cosine transform
     fft(x)

dcost(x)
```

error.bar *Plot Error Bar*

Description

Plot Error Bar

Usage

```
error.bar(x, y, lo, hi, pch = 1, col = 1, barw = 0.1, add = FALSE, ...)
```

Arguments

x	X-values
y	Y-values
lo	Lower limit of error bars
hi	Upper limit of error bars
pch	plotting character
col	color
barw	width of the bar
add	logical, add=FALSE starts a new plot
...	other plotting parameters

Value

graphical side effects

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

Examples

```
x = 1:10
y = 2*x+5
zup = rnorm(10)


zup = zup-min(zup)+.5
zdown = rnorm(10)
zdown = zdown-min(zdown)+.2


####  example with same error on either side:
error.bar(x, y, y-zup, y+zup, pch = 1, col = 'brown' , barw = 0.1, add =
FALSE)


####  example with different  error on either side:
error.bar(x, y, y-zdown, y+zup, pch = 1, col = 'brown' , barw = 0.1, add
= FALSE)
```

`flipGSVD` *Flip output of GSVD*

Description

Flip (reverse order) output of GSVD

Usage

```
flipGSVD(vs, d1 = c(50, 50), d2 = c(48, 50))
```

Arguments

`vs`	list output of GSVD
`d1`	dimensionals of A
`d2`	dimensions of B

Details

This flipping of the matrix is done to agree with the Matlab code.

Value

Same as GSVD, but order of eigenvectors is reversed.

`U`	m by m orthogonal matrix
`V`	p by p orthogonal matrix, p=rank(B)
`X`	n by n nonsingular matrix
`C`	singular values, m by n matrix with diagonal elements shifted from main diagonal
`S`	singular values, p by n diagonal matrix

Note

The GSVD routines are from LAPACK.

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

See Also

GSVD

Examples

```
set.seed(12)

n <- 5
A <- matrix(runif(n*n),nrow=n)
B <- matrix(runif(n*n),nrow=n)

VS = GSVD(A, B)

FVS = flipGSVD(VS, d1 = dim(A) , d2 = dim(B) )
## see that order of eigen vectors is reversed
diag(VS$S)
diag(FVS$S)
```

`gcval` *Get c-val*

Description

Extract the smallest regularization parameter.

Usage

```
gcval(U, s, b, npoints)
```

Arguments

`U`	U matrix from gsvd(G, L)
`s`	[diag(C) diag(S)] which are the lambdas and mus from the gsvd
`b`	the data to try and match
`npoints`	number of alphas to estimate

Details

Evaluate the GCV function gcv_function at npoints points.

Value

List:

`reg_min`	alpha with the minimal g (scalar)
`g`	|| Gm_(alpha,L) - d ||^2 / (Tr(I - GG#)^2
`alpha`	alpha for the corresponding g

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

See Also

gcv_function

Examples

```
set.seed(2015)
VSP = vspprofile()
t = VSP$t2
G = VSP$G
M = VSP$M
N = VSP$N

L1 = get_l_rough(N,1);
littleU = PEIP::GSVD(as.matrix(G), as.matrix(L1) );

BIGU = flipGSVD(littleU, dim(G), dim(L1) )

U1 = BIGU$U
V1 =BIGU$V
X1=BIGU$X
Lam1=BIGU$C
M1=BIGU$S

lam=sqrt(diag(t(Lam1 %*% Lam1)));

mu=sqrt(diag(t(M1)%*%M1));

p=rnk(L1);

sm1=cbind(lam[1:p],mu[1:p])

### % get the gcv values varying alpha

###
ngcvpoints=1000;

HI = gcval(U1,sm1,t,ngcvpoints);
```

---

gcv_function *gcv func*

Description

Auxiliary routine for GCV calculations

Usage

```
gcv_function(alpha, gamma2, beta)
```

Arguments

`alpha`	parameter
`gamma2`	square of the gamma from the gsvd
`beta`	projected data to fit

Value

vector, g - || Gm_(alpha,L) - d ||^2 / (Tr(I - GG#)^2

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

References

Aster, R.C., C.H. Thurber, and B. Borchers, *Parameter Estimation and Inverse Problems*, Elsevier Academic Press, Amsterdam, 2005.

`get_l_rough` *One-D Roughening*

Description

Returns a 1D differentiating matrix operating on a series with n points.

Usage

```
get_l_rough(n, deg)
```

Arguments

`n`	integer, number of data points
`deg`	order of the derivative to approximate

Details

Used to get first and 2nd order roughening matrices for 1-D problems

Value

Matrix:discrete differentiation matrix

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

References

Aster, R.C., C.H. Thurber, and B. Borchers, *Parameter Estimation and Inverse Problems*, Elsevier Academic Press, Amsterdam, 2005.

Examples

```
### first order roughening matrix for a 10 by 10 model: a sparse matrix
N = 10
L1 = get_l_rough(10,1);

### second order roughening matrix for a 10 by 10 model
N = 10
L2 = get_l_rough(10,2);
```

---

ginv *Get inverse*

Description

Get inverse of matrisx or solve Ax=b.

Usage

```
ginv(G, x, tol = 1e-12)
```

Arguments

`G`	Design Matrix
`x`	right hand side
`tol`	tolerance

Details

This function used as alternative to matlab code that does this h = G\x to get the inverse

Value

inverse as a N by 1 matrix.

Note

Be careful about the usage of tolerance

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

See Also

solve, Ainv

Examples

```
set.seed(2015)
GAB = matrix(runif(36), ncol=6)
truex =rnorm(ncol(GAB))
rhs = GAB %*% truex

rhs = as.vector(rhs )

tout = ginv(GAB, rhs, tol = 1e-12)
tout - truex
```

GSVD *Generalized SVD*

Description

Wrapper for generalized svd from LAPACK

Usage

```
GSVD(A, B)
```

Arguments

A	Matrix, see below
B	Matrix, see below

Details

The A and B matrices will be, A=U*C*t(X) and B=V*S*t(X), respectively.

Since PEIP is based on a book, which is iteslef based on MATLAB routines, the convention here follows the book. The R implementation uses LAPACK and wraps the function so the output will comply with the book. See page 104 of the second edition of the Aster book cited below. That said, the purpose is to find an inversion of the form Y = t(A aB), where a is a regularization parameter, B is smoothing matrix and A is the design matrix for the forward problem. The input matrices A and B are assumed to have full rank, and p = rank(B). The generalized singular values are then gamma = lambda/mu, where lambda = sqrt(diag(t(C)*C)) and mu = sqrt(diag(t(S)*S)).

Value

U	m by m orthogonal matrix
V	p by p orthogonal matrix, p=rank(B)
X	n by n nonsingular matrix
C	singular values, m by n matrix with diagonal elements shifted from main diagonal
S	singular values, p by n diagonal matrix

Note

Requires R version of LAPACK. The code is a wrapper for the dggsvd function in LAPACK. The author thanks Berend Hasselman for advice and help preparing this function.

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

References

Aster, R.C., C.H. Thurber, and B. Borchers, *Parameter Estimation and Inverse Problems*, Elsevier Academic Press, Amsterdam, 2005.

See Also

flipGSVD

Examples

```
# Example from NAG F08VAF

A <- matrix(1:15, nrow=5,ncol=3)
 B <- matrix(c(8,1,6,
              3,5,7,
              4,9,2), nrow=3, byrow=TRUE)

z <- GSVD(A,B)
C <- z$C
S <- z$S
sqrt(diag(t(C) %*% C)) / sqrt(diag(t(S) %*% S))
testA = A - z$U %*% C %*% t(z$X)
testB = B - z$V %*% S %*% t(z$X)

print(testA)
print(testB)
```

idcost *Inverse cosine transform*

Description

Takes the column-by-column inverse discrete cosine transform of Y.

Usage

```
idcost(Y)
```

Arguments

Y	Input cosine transform

Value

Time series

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

References

Aster, R.C., C.H. Thurber, and B. Borchers, *Parameter Estimation and Inverse Problems*, Elsevier Academic Press, Amsterdam, 2005.

See Also

dcost

Examples

```
x <- 1:4

### compare fft with cosine transform
     fft(x)

zig = dcost(x)
zag = idcost(zig)
```

imagesc *Image Display*

Description

Display image in matlab format, i.e. flip and transpose.

Usage

```
imagesc(G, col = grey((1:99)/100), ...)
contoursc(G, ...)
```

Arguments

G	Image matrix
col	color scale
...	graphical parameters

Details

Program flips image and transposes prior to plotting. The contour version does the same and can be used to add contours.

Value

graphical side effects

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

Examples

```
mtruem=matrix(rep(0, 16*16), ncol=16,nrow=16);

mtruem[9,9]=1; mtruem[9,10]=1; mtruem[9,11]=1;
mtruem[10,9]=1; mtruem[10,11]=1;
mtruem[11,9]=1; mtruem[11,10]=1; mtruem[11,11]=1;
mtruem[2,3]=1; mtruem[2,4]=1;
mtruem[3,3]=1; mtruem[3,4]=1;

imagesc(mtruem, asp=1)
```

interp2grid *Bilinear and Bicubic Interpolation to Grid*

Description

This code includes a bicubic interpolation and a bilinear interpolation adapted from Numerical Recipes in C: The art of scientific computing (chapter 3... bicubic interpolation) and a bicubic interpolation from in java code.

Inputs are a list of points to interpolate to and from raster objects of class 'asc' (adehabitat package), 'RasterLayer' (raster package) or 'SpatialGridDataFrame' (sp package).

Usage

```
interp2grid(mat,xout,yout,xin=NULL,yin=NULL,type=2)
```

Arguments

mat	a matrix of data that can be a raster matrix of class 'asc' (adehabitat package), 'RasterLayer' (raster package) or 'SpatialGridDataFrame' (sp package) NA values are not permitted.. data must be complete.
xout	a vector of data representing x coordinates of the output grid. Resulting grid must have square cell sizes if mat is of class 'asc', 'RasterLayer' or 'SpatialGridDataFrame'.
yout	a vector of data representing x coordinates of the output grid. Resulting grid must have square cell sizes if mat is of class 'asc', 'RasterLayer' or 'SpatialGridDataFrame'.
xin	a vector identifying the locations of the columns of the input data matrix. These are automatically populated if mat is of class 'asc', 'RasterLayer' or 'SpatialGridDataFrame'.
yin	a vector identifying the locations of the rows of the input data matrix. These are automatically populated if mat is of class 'asc', 'RasterLayer' or 'SpatialGridDataFrame'.
type	an integer value representing the type of interpolation method used. 1 - bilinear adapted from Numerical Recipes in C 2 - bicubic adapted from Numerical Recipes in C 3 - bicubic adapted from online java code

Value

Returns a matrix of the originating class.

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

Examples

```
tx = seq(0,3,0.1)
ty = seq(0,3,0.1)

     tmat = matrix(runif(16,1,16),nrow=4)
     txin = seq(0,3,length=4)
     tyin = seq(0,3,length=4)

     bilinear1 = interp2grid(tmat,tx,ty,txin, tyin,    type=1)
     bicubic2 = interp2grid(tmat,tx,ty,txin, tyin, type=2)
     bicubic3 = interp2grid(tmat,tx,ty,txin, tyin, type=3)

    par(mfrow=c(2,2),cex=1)
             image(tmat,main='base',zlim=c(0,16),col=heat.colors(100))
             image(bilinear1,main='bilinear',zlim=c(0,16),col=heat.colors(100))
             image(bicubic2,main='bicubic2',zlim=c(0,16),col=heat.colors(100))
             image(bicubic3,main='bicubic3',zlim=c(0,16),col=heat.colors(100))
```

---

irls *Iteratively reweight least squares*

Description

Uses the iteratively reweight least squares strategy to find an approximate L_p solution to Ax=b.

Usage

```
irls(A, b, tolr, tolx, p, maxiter)
```

Arguments

A	Matrix of the system of equations.
b	Right hand side of the system of equations
tolr	Tolerance below which residuals are ignored
tolx	Stopping tolerance. Stop when (norm(newx-x)/(1+norm(x)) < tolx)
p	Specifies which p-norm to use (most often, p=1.)
maxiter	Limit on number of iterations of IRLS

Details

Use to get L-1 norm solution of inverse problems.

Value

x	Approximate L_p solution

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

References

Aster, R.C., C.H. Thurber, and B. Borchers, *Parameter Estimation and Inverse Problems*, Elsevier Academic Press, Amsterdam, 2005.

Examples

```
t = 1:10
y=c(109.3827,187.5385,267.5319,331.8753,386.0535,
428.4271,452.1644,498.1461,512.3499,512.9753)
sigma = rep(8, length(y))
N=length(t);

### % Introduce the outlier
y[4]=y[4]-200;

G = cbind( rep(1, N), t, -1/2*t^2 )

### % Apply the weighting

yw = y/sigma;

Gw = G/sigma

m2 = solve( t(Gw) %*% Gw , t(Gw) %*% yw, tol=1e-12 )


###  Solve for the 1-norm solution

m1 = irls(Gw,yw,1.0e-5,1.0e-5,1,25)
m1
```

`irlsl1reg` *L1 least squares with sparsity*

Description

Solves the system Gm=d using sparsity regularization on Lm. Solves the L1 regularized least squares problem: min norm(G*m-d,2)^2+alpha*norm(L*m,1)

Usage

```
irlsl1reg(G, d, L, alpha, maxiter = 100, tolx = 1e-04, tolr = 1e-06)
```

Arguments

G	design matrix
d	right hand side
L	regularization matrix
alpha	regularization parameter
maxiter	Maximum number of IRLS iterations
tolx	Tolerance on successive iterates
tolr	Tolerance below which we consider an element of L*m to be effectively zero

Value

m	model vector

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

References

Aster, R.C., C.H. Thurber, and B. Borchers, *Parameter Estimation and Inverse Problems*, Elsevier Academic Press, Amsterdam, 2005.

Examples

```
n = 20
G = shawG(n,n)

spike = rep(0,n)
spike[10] = 1

spiken = G %*%  spike

wts = rep(1, n)
delta = 1e-03
set.seed(2015)
dspiken = spiken + 6e-6 *rnorm(length(spiken))
L1 = get_l_rough(n,1);
alpha = 0.001

k = irlsl1reg(G, dspiken, L1, alpha, maxiter = 100, tolx = 1e-04, tolr = 1e-06)


plotconst(k,-pi/2,pi/2, ylim=c(-.2,  0.5), xlab="theta", ylab="Intensity" );
```

kac *Kaczmarz*

Description

Implements Kaczmarz's algorithm to solve a system of equations iteratively

Usage

```
kac(A, b, tolx, maxiter)
```

Arguments

A	Constraint matrix
b	right hand side
tolx	difference tolerence for successive iterations (stopping criteria)
maxiter	maximum iterations (stopping criteria)

Value

x	solution

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

References

Aster, R.C., C.H. Thurber, and B. Borchers, *Parameter Estimation and Inverse Problems*, Elsevier Academic Press, Amsterdam, 2005.

Examples

```
set.seed(2015)
G = setDesignG()
### % Setup the true model.
mtruem=matrix(rep(0, 16*16), ncol=16,nrow=16);

mtruem[9,9]=1; mtruem[9,10]=1; mtruem[9,11]=1;
mtruem[10,9]=1; mtruem[10,11]=1;
mtruem[11,9]=1; mtruem[11,10]=1; mtruem[11,11]=1;
mtruem[2,3]=1; mtruem[2,4]=1;
mtruem[3,3]=1; mtruem[3,4]=1;

### % reshape the true model to be a vector
mtruev=as.vector(mtruem);
```

```
### % Compute the data.
dtrue=G %*% mtruev;

### % Add the noise.

d=dtrue+0.1*rnorm(length(dtrue));

mkac<-kac(G,d,0.0,200)
par(mfrow=c(1,2))
imagesc(matrix(mtruem,16,16) , asp=1 , main="True Model" );

imagesc(matrix(mkac,16,16) , asp=1 , main="Kacz Solution" );
```

`linesconst` *Plot constant model*

Description

Add to plotting model in piecewise constant form over n subintervals, where n is the length of x.

Usage

```
linesconst(x, l, r, ...)
```

Arguments

`x`	model to be plotted
`l`	left endpoint of plot
`r`	right endpoint of plot
`...`	graphical parameters

Details

Used for plotting vector models

Value

graphical side effects

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

See Also

plotconst

Examples

```
zip = runif(25)
plotconst(zip, 0, 1 )
linesconst(runif(25) , 0, 1 , col='red' )
```

---

lmarq *Lev-Marquardt Inversion*

Description

Use the Levenberg-Marquardt algorithm to minimize f(p)=sum(F_i(p)^2)

Usage

```
lmarq(afun, ajac, p0, tol, maxiter)
```

Arguments

afun	name of the function F(x)
ajac	name of the Jacobian function J(x)
p0	initial guess
tol	stopping tolerance
maxiter	maximum number of iterations allowed

Value

pstar	best solution found.
iter	Iteration count.

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

Examples

```
fun<-function(p){
###  Compute the function values.
fvec=rep(0,length(TM))
fvec=(Q*exp(-D^2*p[1]/(4*p[2]*TM))/(4*pi*p[2]*TM) - H)/SIGMA
  return(fvec)
}
jac <-function( p)
  {
###  use known formula for the derivatives in the Jacobian
    n=length(TM)
```

```
    J= matrix(0,nrow=n,ncol=2)

      J[,1]=(-Q*D^2*exp(-D^2*p[1]/(4*p[2]*TM))/(16*pi*p[2]^2*TM^2))/SIGMA

      J[,2]=(Q/(4*pi*p[2]^2*TM))*
          ((D^2*p[1])/(4*p[2]*TM)-1)*exp(-D^2*p[1]/(4*p[2]*TM))/SIGMA
   return(J)
  }

H=c(0.72, 0.49, 0.30, 0.20, 0.16, 0.12)
TM=c(5.0, 10.0, 20.0, 30.0, 40.0, 50.0)

###  Fixed parameter values.
D=60
Q=50
###  We'll use sigma=1cm.
SIGMA=0.01*rep(1,length(H))
###  The unknown/estimated parameters are S=p(1) and T=p(2).
p0=c(0.001, 1.0)
###  Solve the least squares problem with LM.
 PEST =  lmarq('fun','jac',p0,1.0e-12,100)
```

`loadMAT` *Load a Matlab matfile*

Description

Load a Matlab matfile, rename the internal parameters to get R-objects

Usage

```
loadMAT(fn, pos=1)
```

Arguments

`fn`	file name of MATfile
`pos`	integer, position in search path, default=1

Details

Program reads in previously saved mat-files and extracts the data, and renames the variables to match the book.

Value

Whatever is in the MATfile

Note

Matfiles are created using the matlab2R routines

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

`l_curve_corner` *L Curve Corner*

Description

Retrieve corner of L-curve

Usage

```
l_curve_corner(rho, eta, reg_param)
```

Arguments

`rho`	misfit
`eta`	model norm or seminorm
`reg_param`	regularization parameter

Value

`reg_corner`	the value of reg_param with maximum curvature
`ireg_corner`	the index of the value in reg_param with maximum curvature
`kappa`	the curvature for each reg_param

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

References

Aster, R.C., C.H. Thurber, and B. Borchers, *Parameter Estimation and Inverse Problems*, Elsevier Academic Press, Amsterdam, 2005.

Examples

```
####  Vertical Seismic Profile example
set.seed(2015)
VSP = vspprofile()
t = VSP$t2
G = VSP$G
M = VSP$M
N = VSP$N

L1 = get_l_rough(N,1);
littleU = PEIP::GSVD(as.matrix(G), as.matrix(L1) );

BIGU = flipGSVD(littleU, dim(G), dim(L1) )

U1 = BIGU$U
V1 =BIGU$V
X1=BIGU$X
Lam1=BIGU$C
M1=BIGU$S



K1 =  l_curve_tgsvd(U1,t,X1,Lam1,G,L1);

rho1 =K1$rho
eta1 =K1$eta
reg_param1 =K1$reg_param
m1s =K1$m

### % store where the corner is (from visual inspection)
vcorn = l_curve_corner(rho1, eta1, reg_param1)

ireg_corner1=vcorn$reg_corner
rho_corner1=rho1[ireg_corner1];
eta_corner1=eta1[ireg_corner1];
print(paste('1st order reg corner is:  ',ireg_corner1));


plot(rho1,eta1,type="b", log="xy" , xlim=c(1e-4, 1e-2) , ylim=c(6e-6, 2e-4)  ,
     xlab="Residual Norm ||Gm-d||_2", ylab="Solution Seminorm ||Lm||_2"  );
points(rho_corner1, eta_corner1, col='red', cex=2 )
```

`l_curve_tgsvd` *L curve tgsvd*

Description

L curve parematers and models for truncated gsvd regularization.

Usage

```
l_curve_tgsvd(U, d, X, Lam, G, L)
```

Arguments

U	U, output of GSVD
d	output of GSVD
X	output of GSVD
Lam	output of GSVD
G	output of GSVD
L	output of GSVD

Value

List:

eta	the solution seminorm ||Lm||
rho	the residual norm ||G m - d||
reg_param	corresponding regularization parameters
m	corresponding suite of models for truncated GSVD

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

References

Aster, R.C., C.H. Thurber, and B. Borchers, *Parameter Estimation and Inverse Problems*, Elsevier Academic Press, Amsterdam, 2005.

Examples

```
####  Vertical Seismic Profile example
set.seed(2015)
VSP = vspprofile()
t = VSP$t2
G = VSP$G
M = VSP$M
N = VSP$N


L1 = get_l_rough(N,1);
littleU = PEIP::GSVD(as.matrix(G), as.matrix(L1) );

BIGU = flipGSVD(littleU, dim(G), dim(L1) )
```

```
U1 = BIGU$U
V1 =BIGU$V
X1=BIGU$X
Lam1=BIGU$C
M1=BIGU$S



K1 =  l_curve_tgsvd(U1,t,X1,Lam1,G,L1);

rho1 =K1$rho
eta1 =K1$eta
reg_param1 =K1$reg_param
m1s =K1$m


### % store where the corner is (from visual inspection)
ireg_corner1=8;
rho_corner1=rho1[ireg_corner1];
eta_corner1=eta1[ireg_corner1];
print(paste('1st order reg corner is:  ',ireg_corner1));


plot(rho1,eta1,type="b", log="xy", xlim=c(1e-4, 1e-2) , ylim=c(6e-6, 2e-4) ,
     xlab="Residual Norm ||Gm-d||_2", ylab="Solution Seminorm ||Lm||_2"  );
```

`l_curve_tikh_gsvd` *L-curve tikh gsvd*

Description

L-curve tikh gsvd

Usage

```
l_curve_tikh_gsvd(U, d, X, Lam, Mu, G, L, npoints, varargin = NULL)
```

Arguments

`U`	from the gsvd
`d`	data vector for the problem G*m=d
`X`	from the gsvd
`Lam`	from the gsvd
`Mu`	from the gsvd

G	system matrix
L	roughening matrix
npoints	Number of points
varargin	alpha_min, alpha_max: if specified, constrain the logrithmically spaced regularization parameter range, otherwise an attempt is made to estimate them from the range of generalized singular values

Details

Uses output of GSVD

Value

eta	- the solution seminorm ||Lm||
rho	- the residual norm ||G m - d||
reg_param	- corresponding regularization parameters
m	- corresponding suite of models for truncated GSVD

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

Examples

```
####  Vertical Seismic Profile example
set.seed(2015)
VSP = vspprofile()
t = VSP$t2
G = VSP$G
M = VSP$M
N = VSP$N

L1 = get_l_rough(N,1);
littleU = PEIP::GSVD(as.matrix(G), as.matrix(L1) );

BIGU = flipGSVD(littleU, dim(G), dim(L1) )

U1 = BIGU$U
V1 =BIGU$V
X1=BIGU$X
Lam1=BIGU$C
M1=BIGU$S

K1 =  l_curve_tikh_gsvd(U1,t,X1,Lam1,M1, G,L1, 25);

rho1 =K1$rho
eta1 =K1$eta
reg_param1 =K1$reg_param
```

```
m1s =K1$m


###  store where the corner is (from visual inspection)
ireg_corner1=8;
rho_corner1=rho1[ireg_corner1];
eta_corner1=eta1[ireg_corner1];
print(paste('1st order reg corner is:  ',ireg_corner1));


plot(rho1,eta1,type="b", log="xy", xlim=c(1e-4, 1e-2) , ylim=c(6e-6, 2e-4) ,
     xlab="Residual Norm ||Gm-d||_2", ylab="Solution Seminorm ||Lm||_2"  );
```

---

l_curve_tikh_svd *L-curve Tikhonov*

Description

L-curve for Tikhonov regularization

Usage

```
l_curve_tikh_svd(U, s, d, npoints, varargin = NULL)
```

Arguments

U	matrix of data space basis vectors from the svd
s	vector of singular values
d	the data vector
npoints	the number of logarithmically spaced regularization parameters
varargin	alpha_min, alpha_max: if specified, constrain the logrithmically spaced regularization parameter range, otherwise an attempt is made to estimate them from the range of singular values

Details

Calculates the L-curve

Value

eta	the solution norm ||m|| or seminorm ||Lm||
rho	the residual norm ||G m - d||
reg_param	corresponding regularization parameters

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

Examples

```
####  Vertical Seismic Profile example
set.seed(2015)
VSP = vspprofile()
t = VSP$t2
G = VSP$G
M = VSP$M
N = VSP$N

L1 = get_l_rough(N,1);
littleU = PEIP::GSVD(as.matrix(G), as.matrix(L1) );

BIGU = flipGSVD(littleU, dim(G), dim(L1) )

U1 = BIGU$U
V1 =BIGU$V
X1=BIGU$X
Lam1=BIGU$C
M1=BIGU$S

K1 = l_curve_tikh_svd(U1, diag(M1) , X1, 25, varargin = NULL)

rho1 =K1$rho
eta1 =K1$eta
reg_param1 =K1$reg_param
m1s =K1$m

###  store where the corner is (from visual inspection)
ireg_corner1=8;
rho_corner1=rho1[ireg_corner1];
eta_corner1=eta1[ireg_corner1];
print(paste("1st order reg corner is:  ",ireg_corner1));

plot(rho1,eta1,type="b", log="xy" ,
     xlab="Residual Norm ||Gm-d||_2", ylab="Solution Seminorm ||Lm||_2"  );
```

---

mcmc *Maximum likelihood Models*

Description

Maximum likelihood Models

Usage

```
mcmc(alogprior, aloglikelihood, agenerate, alogproposal, m0, niter)
```

Arguments

alogprior	Name of a function that computes the log of the prior distribution.
aloglikelihood	Name of a function the computes the log of the likelihood.
agenerate	Name of a function that generates a random model from the current model using the
alogproposal	Name of a function that computes the log of the proposal distribution r(x,y).
m0	Initial model
niter	Number of iterations to perform

Value

mout	MCMC samples
mMAP	Best model found in the MCMC simulation.
accrate	Acceptance rate

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

Examples

```
fun <-function(m,x)
{
  y=m[1]*exp(m[2]*x)+m[3]*x*exp(m[4]*x)
  return(y)
}

generate <-function( x) {
  y=x+step*rnorm(4)
  return(y)
}

logprior <-function(m)
{
  if( (m[1]>=0) & (m[1]<=2) &
     (m[2]>=-0.9) & (m[2]<=0) &
     (m[3]>=0) & (m[3]<=2) &
     (m[4]>=-0.9) & (m[4]<=0)  )
    {
      lp=0
    }
  else
    {
      lp= -Inf
    }

  return(lp)
}
loglikelihood <-function(m)
```

```
{
  fvec=(y-fun(m,x))/sigma
  L=(-1/2)*sum(fvec^2)
  return(L)
}
logproposal <-function(x,y)
  {
    LR=(-1/2)*sum((x-y)^2/step^2)
    return(LR)
  }

###  Generate the data set.
x=seq(from=1, by=0.25, to=7.0)

mtrue=c(1.0, -0.5, 1.0, -0.75)

ytrue=fun(mtrue,x)

sigma=0.01*rep(1, times= length(ytrue) )

y=ytrue+sigma*rnorm(length(ytrue) )

### set the MCMC parameters
### number of skips to reduce autocorrelation of models
skip=100
### burn-in steps
BURNIN=1000
### number of posterior distribution samples
N=4100
### MVN step size
step = 0.005*rep(1,times=4)

###  We assume flat priors here
m0 = c(0.9003,
   -0.5377,
    0.2137,
   -0.0280)

alogprior='logprior'
aloglikelihood='loglikelihood'
agenerate='generate'
alogproposal='logproposal'

### ###  initialize model at a random point on [-1,1]

###  m0=(runif(4)-0.5)*2
###  this is the matlab initialization:
m0 = c(0.9003,
   -0.5377,
    0.2137,
   -0.0280)

MM =  mcmc('logprior','loglikelihood','generate','logproposal',m0,N)
```

```
mout = MM[[1]]
mMAP= MM[[2]]
pacc= MM[[3]]
```

Mnorm *Matrix Norm*

Description

Matrix Norm

Usage

```
Mnorm(X, k = 2)
```

Arguments

X	matrix
k	norm number

Details

returns the largest singular value of the matrix or vector

Value

Scalar Norm

Note

if k=1, absolute value; k=2 2-norm (rms); k>2, largest singular value.

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

Examples

```
x = runif(10)

Mnorm(x, k = 2)
```

`nnz` *Non-zeros*

Description

Number of non-zero elements in a vector

Usage

```
nnz(h)
```

Arguments

h	vector

Value

integer

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

Examples

```
zip<-rnorm(15)
nnz(zip)
```

`occam` *Occam inversion*

Description

Occam's inversion

Usage

```
occam(afun, ajac, L, d, m0, delta)
```

Arguments

`afun`	character, function handle that computes the forward problem
`ajac`	character, function handle that computes the Jacobian of the forward problem
`L`	regularization matrix
`d`	data that should be fit
`m0`	guess at the model
`delta`	cutoff to use for the discrepancy principle portion

Value

vector, model found

Note

This is a simple brute force way to do the line search. Much more sophisticated methods are available. Note: we've restricted the line search to the range from 1.0e-20 to 1. This seems to work well in practice, but might need to be adjusted for a particular problem.

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

See Also

bayes

`phi` *Integral of Normal Distribution*

Description

normal distribution and returns the value of the integral

Usage

```
phi(x)
```

Arguments

`x`	endpoint of integration (scalar)

Value

value of integral

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

See Also

erf

Examples

```
 x <- 1.0
##   pracma::erf(x)
 phi(x)
phiinv( phi(x) )
```

---

phiinv *Inverse Normal Distribution Integral*

Description

Calculates the inverse normal distribution from the value of the integral

Usage

```
phiinv(x)
```

Arguments

x	endpoint value of integration (scalar)

Value

value of integral (scalar)

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

See Also

phi

Examples

```
 x <- 1.0
##   pracma::erf(x)
 phi(x)
phiinv( phi(x) )
```

---

picard_vals *Picard plot*

Description

Picard plot parameters for subsequent plotting.

Usage

```
picard_vals(U, sm, d)
```

Arguments

U	the U matrix from the SVD or GSVD
sm	singular values in decreasing order, or the GSVD lambdas divided by the mus in decreasing order
d	data to fit, right hand side

Details

The Picard plot is a method of helping to determine regularization schemes.

Value

List:

utd	the columns of U transposed times d
utd_norm	utd./sm

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

See Also

GSVD

Examples

```
####
n = 20
G = shawG(n,n)
spike = rep(0,n)
spike[10] = 1
dspiken = G

set.seed(2015)
dspiken = dspiken + 6e-6 *rnorm(length(dspiken))
Utube=svd(G);
U = Utube$u
V = Utube$v
S = Utube$d
s=Utube$d
 R3 = picard_vals(U,s,dspiken);
utd = R3$utd
utd_norm= R3$utd_norm
###  Produce the Picard plot.

x_ind=1:length(s);
##
plot( range(x_ind) , range(c(s ,abs(utd),abs(utd_norm))),
          type='n',  log='y', xlab="i", ylab="" )
lines(x_ind,s, col='black')
points(x_ind,abs(utd), pch=1, col='red')
points(x_ind,abs(utd_norm), pch=2, col='blue')

title("Picard Plot for Shaw Problem")
```

---

plotconst *Plot constant model*

Description

Plots a model in piecewise constant form over n subintervals, where n is the length of x.

Usage

```
plotconst(x, l, r, ...)
```

Arguments

x	model to be plotted
l	left endpoint of plot

`r`	right endpoint of plot
`...`	graphical parameters

Details

Used for plotting vector models

Value

graphical side effects

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

See Also

linesconst

Examples

```
zip = runif(25)
plotconst(zip, 0, 1 )
linesconst(runif(25) , 0, 1 , col='red' )
```

---

quadlin — *Lagrange multiplier technique*

Description

Quadratic Linearization

Usage

```
quadlin(Q, A, b)
```

Arguments

`Q`	positive definite symmetric matrix
`A`	matrix with linearly independent rows
`b`	data vector

Details

Solves the problem: min (1/2) t(x)*Q*x with Ax = b. using the Lagrange multiplier technique, where Q is assumed to be symmetric and positive definite and the rows of A are linearly independent.

Value

list:

x	vector of solution values
lambda	Lagrange multiplier

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

Examples

```
###%    Radius of the Earth (km)
    Re=6370.8;
rad = 5000
ri=rad/Re;

q=c(1.083221147,  1.757951474)
H = matrix(rep(0, 4), ncol=2, nrow=2)

H[1,1]=1.508616069 - 3.520104161*ri + 2.112062496*ri^2;
H[1,2]=3.173750352 - 7.140938293*ri + 4.080536168*ri^2;
H[2,1]=H[1,2];
H[2,2]=7.023621326 - 15.45196692*ri + 8.584426066*ri^2;
A1 =quadlin(H,t(q), 1.0 );
```

---

rnk   *Rank of Matrix*

Description

Return the rank of a matrix. Not to be confused with the R function rank.

Usage

```
rnk(G, tol = 1e-14)
```

Arguments

`G`	Matrix
`tol`	machine tolerance for small numbers

Details

Number of singular values greater than tol.

Value

integer, number of non-zero singular values

Note

duplicate the matlab function rank

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

See Also

svd

Examples

```
hilbert <- function(n) { i <- 1:n; 1 / outer(i - 1, i, "+") }
X <- hilbert(9)[,1:6]
rnk(X)
```

`setDesignG` *Set a Design Matrix.*

Description

Creata design matrix for simulating a tomographic inversion on a simple grid.

Usage

```
setDesignG()
```

Details

Set up a simple design matrix for tomographic in version. This is used in examples and illustrations of tomographics and matrix inversion methods.

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

Examples

```
G = setDesignG()

### show the 56-th row
g = matrix( G[56,] , ncol=16, nrow=16)
imagesc(g)

## Not run:
### show total coverage
zim = matrix(0 , ncol=16, nrow=16)
for(i in 1:dim(G)[1])
{
g = matrix( G[i,] , ncol=16, nrow=16)
zim =zim + g
}
image(zim)



## End(Not run)
```

---

shawG *Shaw Model of Slit Diffraction*

Description

Creates the design matrix for the Shaw inverse problem of diffraction through a narrow slot.

Usage

```
shawG(m, n)
```

Arguments

m	integer, number of rows
n	integern number of columns

Details

See Aster's book for a details explaination.

Value

Matrix used for creating data and inversion.

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

References

C. B. Shaw, Jr., "Improvements of the resolution of an instrument by numerical solution of an integral equation", J. Math. Anal. Appl. 37: 83-112, 1972.

Examples

```
n = 20
G = shawG(n,n)

spike = rep(0,n)
spike[10] = 1

dspiken = G %*%  spike

plot(dspiken)
```

`sirt` *SIRT Algorithm for sparse matrix inversion*

Description

Row action method for inversion of matrices, using SIRT algorithm.

Usage

```
sirt(A, b, tolx, maxiter)
```

Arguments

`A`	Design Matrix
`b`	vector, Right hand side
`tolx`	numeric, tolerance for stopping
`maxiter`	integer, Maximum iterations

Details

Iterates until conversion

Value

Solution vector

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

References

Lees, J. M. and R. S. Crosson (1989): Tomographic inversion for three-dimensional velocity structure at Mount St. Helens using earthquake data, *J. Geophys. Res.*, 94(B5), 5716-5728.

See Also

art, kac

Examples

```
set.seed(2015)
G = setDesignG()
###  Setup the true model.
mtruem=matrix(rep(0, 16*16), ncol=16,nrow=16);

mtruem[9,9]=1; mtruem[9,10]=1; mtruem[9,11]=1;
mtruem[10,9]=1; mtruem[10,11]=1;
mtruem[11,9]=1; mtruem[11,10]=1; mtruem[11,11]=1;
mtruem[2,3]=1; mtruem[2,4]=1;
mtruem[3,3]=1; mtruem[3,4]=1;

###  reshape the true model to be a vector
mtruev=as.vector(mtruem);

###  Compute the data.
dtrue=G %*% mtruev;

###  Add the noise.

d=dtrue+0.01*rnorm(length(dtrue));

msirt<-sirt(G,d,0.01,200)
par(mfrow=c(1,2))
imagesc(matrix(mtruem,16,16) , asp=1 , main="True Model" );

imagesc(matrix(msirt,16,16) , asp=1 , main="SIRT Solution" );
```

---

tinv *Inverse T-distribution*

Description

Inverse T-distribution, qt

Usage

```
tinv(p, nu)
```

Arguments

`p`	P-value
`nu`	degrees of freedom

Details

Wrapper for qt

Value

Quantile for T-distribution

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

See Also

qt

Examples

```
tinv(.4, 10)
```

`USV` *Singular Value Decomposition*

Description

Singular Value Decomposition

Usage

```
USV(G)
```

Arguments

`G`	Matrix

Details

returns matrices U, S, V according to matlab convention.

Value

list:

U	Matrix
S	Matrix, singular values
V	Matrix

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

See Also

svd

Examples

```
 hilbert <- function(n) { i <- 1:n; 1 / outer(i - 1, i, "+") }
     X <- hilbert(9)[,1:6]

h = USV(X)

print( h$U )
```

Vnorm *Vector 2-Norm*

Description

Vector 2-Norm.

Usage

```
Vnorm(X)
```

Arguments

X	numeric vector

Value

Numeric scale norm

Note

This function is intended to duplicated the matlab function norm.

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

Examples

```
V = Vnorm(rnorm(10))
```

`vspprofile` *Vertical Seismic Profile In 1D*

Description

Example vertical 1-dimensional seismic profile used for setting up examples for inverse theory.

Usage

```
vspprofile(M = 50, N = 50, maxdepth = 1000, deltobs = 20,
noise = 2e-04, M1 = c(9000, -6, 0.001))
```

Arguments

`M`	integer, number of rows in in design matrix G, default=50
`N`	integer, number of columns in design matrix G, default=50
`maxdepth`	Maximum depth of model, default = 1000
`deltobs`	integer, sampling interval in depth, default=20
`noise`	gausian noise multiplier, default=2e-04
`M1`	3-vector, linear model for velocity versus depth model

Details

Vertical seismic profile in 1D dimension used for setting up examples in PEIP. Given a simple velocity profile, defined by input parameter M1 create the travel times and designe matrix used for solving an inverse problem. The velocity model is defined as depth versus velocity, and the function inverts that from the slowness. Any model could be used to replace this model. The default model here is taken from an inversion in the Aster book.

Value

list:

G	M by N design matrix
tee	true travel times from model
t2	travel times with noise added
depth	depth samples of model
vee	velocity at the depths indicated
M	input M
N	input N
maxdepth	input maxdepth
deltobs	input delta observation
noise	input noise
M1	True model used for depth versus velocity

Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

References

Aster, R.C., C.H. Thurber, and B. Borchers, *Parameter Estimation and Inverse Problems*, Elsevier Academic Press, Amsterdam, 2005.

Examples

```
V = vspprofile()
### plot quadratic velocity profile
plot(V$vee, -V$depth, main="VSP: velocity increasing with depth")
dobs = seq(from=V$deltobs, to=V$maxdepth, by=V$deltobs)
### plotdepth versus time (not linear)
plot(dobs, V$t2)
abline(lm(V$t2 ~ dobs) )
```

Index