Package ‘curve’

September 8, 2005

Title Dynamic Systems Estimation - Curvature extensions

Description Multivariate Time Series - extensions

Depends R (>= 2.0.0), setRNG (>= 2004.4-1), tframe (>= 2005.9-1), dse1 (>= 2005.9-1)

Version 2005.9-1

LazyLoad yes

License Free. See the LICENCE file for details.

Author Paul Gilbert <pgilbert@bank-banque-canada.ca>

Maintainer Paul Gilbert <pgilbert@bank-banque-canada.ca>

URL http://www.bank-banque-canada.ca/pgilbert

R topics documented:

curvature *Curvature*

Description

Curvature calculations and summary statistics as in Bates and Watts.

Usage

```
curvature(func, ...)
## Default S3 method:
curvature(func, x, func.args=NULL, d=0.01, eps=1e-4,r=6,
    signif=0.05, show.details=FALSE, warn=TRUE, ...)
## S3 method for class 'Darray':
curvature(func, signif = 0.05,
  show.extra.details=FALSE, show.details=show.extra.details, warn=TRUE, ...)
```

Arguments

`func`	a function for the default method, a Darray object, or a object for which a specific method is defined (e.g. a TSestModel).
`...`	arguments to be passed to other methods.
`x`	parameters to the function.
`func.args`	list with additional argument to function `func`.
`d`	The fraction of x to use for the initial numerical approximation.
`eps`	Used instead of d for elements of x which are zero.
`r`	The number of Richardson improvement iterations.
`signif`	The significance level for F test (passed to `qf`).
`show.details`	logical indicating if intermediate calculations should be printed.
`show.extra.details`	logical indicating if extra intermediate calculations should be printed.
`warn`	see `effectiveCurvature`.

Details

This function is generic. It can be called with a function and parameter x, in which case the Bates and Watts D matrix is calculated, or with a previously calculated D matrix. A function `func` should return a sample space vector at the parameter value x. The method for a `Darray` object works on the result from genD which has already done most of the calculations. The Darray has an element Dlist with the 3 elements as follows: D is a matrix of first(gradients) and second order partial derivatives organized in the same manner as Bates and Watts. (first p columns are the gradients and the next p(p-1)/2 columns are the lower triangle of the Hessian). p is the dimension of the parameter space=dim of the tangent space. f0 is the function value at the point where the matrix D was calculated. (The calculation should not/does not? depend on this value - but it should be the right dimension and 0's do not work.

Value

A list is returned (with invisible). Curvature summary statistics as in Bates and Watts are in the element stat. A representation of the Bates and Watts D matrix is in the element Dlist. The curvature array C as in Batts and Watts defn. (7.16) p242 and examples p244 & p245 is in the elements C.parameter and C.intrinsic.

References

Bates and Watts(1983), 'Nonlinear Regression Analysis and Its Applications.'

See Also

`genD` `curvature.TSestModel` `effectiveCurvature`

Examples

```
    func <- function(x){c(x[1], x[1], x[2]^2)}
 #   curvature(func, c(2,2))
```

`curvatureStats` *Curvature Statistics Utility Used by Curvature and Project*

Description

xxx

Usage

```
curvatureStats(cur, n, signif=0.05)
```

Arguments

`cur`	the relative curvature array. See `relCurvature`.
`n`	`n-p` is denominator degrees of freedom for F statistic, where `p` is `ncol(cur)`.
`signif`	The significance level for F test (passed as `1 - signif` to `qf`).

Details

...

Value

x

References

Bates and Watts(1983), 'Nonlinear Regression Analysis and Its Applications.'

See Also

`curvature` `project` `effectiveCurvature` `curvature.Darray` `relCurvature`

curvature.TSestModel

Curvature for a TSestModel

Description

Calculate curvature for a TSestModel.

Usage

```
## S3 method for class 'TSestModel':
curvature(func, x=coef(func),
 func.args=list(Shape=TSmodel(func), data=TSdata(func)),
  d=0.01, eps=1e-4, r=6, warn=TRUE, compiled=TRUE, ...)
```

Arguments

`func`	a TSestModel object which is used as a function mapping coefficients (parameters) to residuals.
`x`	parameter vector first argument to function func indicating the point with respect to which the derivative is calculated.
`func.args`	list with additional argument to function `func`.
`d`	The fraction of x to use for the initial numerical approximation.
`eps`	Used instead of d for elements of x which are zero.
`r`	The number of Richardson improvement iterations.
`warn`	logical. see `effectiveCurvature`.
`compiled`	logical, set FALSE to use curvature.default (for debugging).
`...`	arguments passed to other methods (currently ignored).

Details

See the generic version of the function.

See Also

`curvature` `effectiveCurvature`

Examples

```
if(is.R()) data("eg1.DSE.data.diff", package="dse1")
model <- estVARXls(TSdata( output=outputData(eg1.DSE.data.diff, series=1:2)),
                   max.lag=2)
curvature(genD(model))
```

`effectiveCurvature` *Effective Curvature Utilitiy Used by Curvature and Project*

Description

xxx

Usage

```
effectiveCurvature(cur, QRofD, residual, s.sqr, show.details=FALSE,
warn=TRUE)
```

Arguments

`cur`	the relative curvature array. See `relCurvature`.
`QRofD`	QR decomposition of D array from Bates and Watts.
`residual`	point in sample space where the curvature should be calculated (possibly shifted by subtracting actual data to give a residual).
`s.sqr`	sample estimate of the residual variance.
`show.details`	logical indicating if intermediate calculations should be printed.
`warn`	see details.

Details

Effective residual curvature from Bates and Watts p260 Calculate the scaled RMS curvatures relative to a confidence disk radius and extreme axis ratios. ref. Bates and Watts (1983) p254 eqn (7.23). and Bates and Watts J.R. Statist.Soc. B (1980).

Transform the residual vector by multiply Q-transpose and sqrt(s.sqr*p). Calculate the p by p effective residual curvature matrix B and its eigenvalues. Bates and Watts p260

... If I-B is not positive definite, where B is the effective residual curvature matrix, and warn is TRUE, then a warning will indicate that the calculation does not seem to correspond to a local minimum.

Value

x

References

Bates and Watts(1980), J.R. Statist.Soc. B. Bates and Watts(1983), 'Nonlinear Regression Analysis and Its Applications.'

See Also

`curvature` `project` `curvatureStats` `relCurvature` `genD` *Generate Bates and Watts D Matrix*

Description

Generate a matrix of function derivative information.

Usage

```
genD(func, x, func.args=NULL, d=0.01, eps=1e-4, r=6)
## Default S3 method:
genD(func, x, func.args=NULL, d=0.01, eps=1e-4, r=6)
```

Arguments

`func`	a function for which the first (vector) argument is used as a parameter vector.
`...`	arguments to be passed to other methods.
`x`	The parameter vector first argument to `func`.
`d`	The fraction of x to use for the initial numerical approximation.
`eps`	Used instead of d for elements of x which are zero.
`r`	The number of Richardson improvement iterations.
`func.args`	list with additional argument to function `func`.

Details

The derivatives are calculated numerically using Richardson improvement. `genD.default` calculates a numerical approximation of the first and second derivatives (gradient and hessian) of `func` at the point x. The calculation is done by Richardson's extrapolation (see e.g.Linfield and Penny The method should be used when accuracy, as opposed to speed, is important. The size of d is iteratively reduced and the Richardson algorithm is applied to improve the accuracy of the computation.

Value

A list with elements as follows: D is a matrix of first(gradients) and second order partial derivatives organized in the same manner as Bates and Watts. (The first p columns are the gradients and the next p(p-1)/2 columns are the lower triangle of the Hessian). p is the dimension of the parameter space=dim of the tangent space. f0 is the function value at the point where the matrix D was calculated. Also the arguments func, x, d, eps, r are returned in the list.

References

G.R.Linfield and J.E.T.Penny,"Microcomputers in Numerical Analysis"

See Also

`curvature` `genD.TSestModel` `gradNumerical`

Examples

```
func <- function(x){c(x[1], x[1], x[2]^2)}
z <- genD(func, c(2,2,5))
```

genD.TSestModel *Generate Bates and Watts D Matrix*

Description

Generate a matrix of function derivative information.

Usage

```
## S3 method for class 'TSestModel':
genD(func, x=coef(func),
    func.args=list(Shape=TSmodel(func), data=TSdata(func)),
    d=0.01, eps=1e-4, r=6 )
## S3 method for class 'ARMA':
genD(func, x=coef(func),
    func.args=list(Shape=TSmodel(func), data=TSdata(func)),
    d=0.01, eps=1e-4, r=6)
## S3 method for class 'innov':
genD(func, x=coef(func),
    func.args=list(Shape=TSmodel(func), data=TSdata(func)),
    d=0.01, eps=1e-4, r=6)
```

Arguments

`func`	a TSestModel or TSmodel object which is used as a function mapping coefficients (parameters) to residuals.
`x`	parameter vector first argument to function func indicating the point with respect to which the derivative is calculated.
`func.args`	list with additional argument to function `func`.
`d`	The fraction of x to use for the initial numerical approximation.
`eps`	Used instead of d for elements of x which are zero.
`r`	The number of Richardson improvement iterations.

Details

The derivatives are calculated numerically using Richardson improvement.

Value

A list with three elements as follows: D is a matrix of first(gradients) and second order partial derivatives organized in the same manner as Bates and Watts. (The first p columns are the gradients and the next p(p-1)/2 columns are the lower triangle of the Hessian). p is the dimension of the parameter space=dim of the tangent space. f0 is the function value at the point where the matrix D was calculated.

See Also

`genD` `curvature`

Examples

```
if(is.R()) data("eg1.DSE.data.diff", package="dse1")
model <- estVARXls(TSdata( output=outputData(eg1.DSE.data.diff, series=1:2)),
                          max.lag=2)
z <- genD(model)
```

`gradNumerical` *Gradient of a Function*

Description

Calculate the gradient of a function by simple epsilon differencing.

Usage

```
gradNumerical(func,x, eps=1e-12)
```

Arguments

`func`	a function with a real result.
`x`	a real or real vector argument to func, indicating the point at which the gradient is to be calculated.
`eps`	The epsilon difference to use for calculating the gradient.

Value

A real or real vector of the calculated gradient.

See Also

`gradRichardson`

Examples

```
gradNumerical(sin, pi)
```

`gradRichardson` *Gradient of a Function*

Description

Calculate the gradient of a function by Richardson improvement.

Usage

```
gradRichardson(func, x, d=0.01, eps=1e-4, r=6, show.details=FALSE)
```

Arguments

`func`	A function with a real result.
`x`	A real or real vector argument to func, indicating the point at which the gradient is to be calculated.
`d`	the fraction of x to use for the initial numerical approximation. The default is set to 0.01*x or eps if x is 0.0.
`eps`	Used instead of d for elements of x which are zero.
`r`	The number of Richardson improvement iterations (repetions with successly smaller d).
`show.details`	logical indicating if detailed calculations should be shown.

Details

This function calculates a numerical approximation of the first derivative of func at the point x. The calculation is done by Richardson's extrapolation (see eg. Linfield and Penny he method should be used if accuracy, as opposed to speed, is important.

GENERAL APPROACH – GIVEN THE FOLLOWING INITIAL VALUES: INTERVAL VALUE D, NUMBER OF ITERATIONS R, AND REDUCED FACTOR V. - THE FIRST ORDER aproximation to the DERIVATIVE WITH RESPECT TO Xi IS

F'(Xi)=F(X1,...,Xi+D,...,Xn) - F(X1,...,Xi-D,...,Xn)/(2*D)

– REPEAT r TIMES with successively smaller D and then apply Richardson extraplolation

Value

A real or real vector of the calculated gradient.

References

G.R.Linfield and J.E.T.Penny,"Microcomputers in Numerical Analysis"

See Also

`gradNumerical`

Examples

```
gradRichardson(sin, pi)
```

`hessian` *Calculate Hessian Matrix*

Description

Calculatate the hessian matrix of a function at a parameter value.

Usage

```
    hessian(func, x, func.args=NULL, d=0.01, eps=1e-4, r=6)
    ## Default S3 method:
    hessian(func, x, func.args=NULL, d=0.01, eps=1e-4, r=6)
```

Arguments

`func`	a function for which the first (vector) argument is used as a parameter vector.
`x`	The parameter vector first argument to func.
`d`	The fraction of x to use for the initial numerical approximation.
`eps`	Used instead of d for elements of x which are zero.
`r`	The number of Richardson improvement iterations.
`func.args`	list with additional argument to function `func`.

Details

This function uses genD and extracts the Hessian matrix..

Value

The Hessian matrix of the function calculated at the point x.

See Also

`hessian.TSestModel` `genD` `span`

`hessian.TSestModel` *Calculate Hessian Matrix*

Description

Calculatate the hessian matrix of a TSmodel at a parameter value.

Usage

```
## S3 method for class 'TSestModel':
hessian(func, x=coef(func),
    func.args=list(Shape=TSmodel(func), data=TSdata(func)),
    d=0.01, eps=1e-4, r=6)
```

Arguments

`func`	a TSestModel object which is used as a function mapping coefficients (parameters) to likelihood.
`x`	The parameter point at which the hessian is calculated.
`func.args`	list with additional argument to the TSmodel evaluation.
`d`	The fraction of x to use for the initial numerical approximation.
`eps`	Used instead of d for elements of x which are zero.
`r`	The number of Richardson improvement iterations.

Details

This function calculates the second derivative of the likelihood for the model at its specified parameter value using given data.

Value

a matrix of second derivative of the likelihood (Fisher Information).

See Also

`hessian` `genD` `span`

`print.curvature` *Specific Methods for Print*

Description

See the generic function description.

Usage

```
## S3 method for class 'curvature':
print(x, ...)
## S3 method for class 'curvatureArray':
print(x, ...)
```

Arguments

`x`	a curvature object to be printed.
`...`	arguments to be passed to other methods.

See Also

`print`

`project` *Project*

Description

project

Usage

```
project(c1, c2, signif = 0.05, eps=1e-5, fuzz=1e-14,
    show.details=FALSE, warn=TRUE)
```

Arguments

`c1`	curvature summary. See details.
`c2`	curvature summary. See details.
`signif`	passed to `curvatureStats`.
`eps`	significance tolerance for singular values. See details.
`fuzz`	tolerance for comparing parameter points. See details.
`show.details`	logical indicating if detailed calculations should be shown.
`warn`	see `effectiveCurvature`.

Details

Under Development

c1 and c2 should be curvature summary statistics as returned by curvature. It is assummed that c1 is a submodel of c2. The tangent space (parameter effects curvature) of the sub-model c1 is a subspace of the tangent space of the larger model. The acceleration space (first normal space, intrinsic curvature effects) of the sub-model is a subspace of the direct sum of the tangent and acceleration spaces of the larger model, so the intrinsic Tangent and acceleration vectors of the submodel c1 can be projected onto the tangent and acceration spaces of the larger model c2. These are called T1inT2, N1inT2, and N1inN2 (T for tangent, N for normal). A second projection (restriction) of c2 onto the tangent and acceleration spaces of c1 is less interesting but may be a useful check. The intrinsic curvature of the larger model should also be intrinsic on the sub-model and the parameter effects of the larger model may be partly intrinsic on the sub-model. These two projected onto the intrinsic curvature space of the submodel (N2andT2inN1) should compare with the intrinsic curvature of the submodel.

Singular values smaller than eps times the largest singular value are considered to be zero when calculating the dimension of the tangent space.

If the parameter points for the two models are not within `fuzz` then a warning is issued to indicate that they do not represent the same point in parameter space.

Value

xxx

See Also

`curvature` `effectiveCurvature`

`relCurvature` *Relative Curvature Utility Used by Curvature and Project*

Description

xxx

Usage

```
relCurvature(s.sqr, R11, R2, show.extra.details=FALSE,
      eps=sqrt(.Machine$double.eps))
```

Arguments

`s.sqr`	sample estimate of the residual variance.
`R11`	submatrix of R. See details.
`R2`	submatrix of R. See details.
`show.extra.details`	logical indicating if extra intermediate calculations should be printed.
`eps`	singular values smaller than eps times the largest singular value are considered to be zero.

Details

The result is the relative curvature array Bates & Watts p242-244 eqn. (7.16) and examples p244-245.

R11 is a p by p sub matrix of R. See Bates & Watts p236 and R2 is the m by pp sub matrix R12 R22 ... If I-B is not positive definite, where B is the effective residual curvature matrix, and warn is TRUE, then a warning will indicate that the calculation does not seem to correspond to a local minimum.

Value

relative curvature array

See Also

`curvature` `project` `effectiveCurvature` `curvatureStats`

`span` *Calculate Span of Tangent Plane*

Description

Calculate the dimension of the tangent space

Usage

```
span(func, x, func.args=NULL, d=0.01, eps=1e-4, r=6,
  show.details=FALSE, ...)
## Default S3 method:
span(func, x, func.args=NULL, d=0.01, eps=1e-4, r=6,
  show.details=FALSE, ...)
```

Arguments

`func`	a function which returns the residual vector for a given parameter vector.
`...`	arguments passed to other methods.
`x`	parameter vector first argument to function func indicating the point with respect to which the derivative is calculated.
`func.args`	list with additional argument to function `func`.
`d`	the fraction of x to use for the initial numerical approximation. The default is set to 0.01*x or eps if x is 0.0.
`eps`	Used instead of d for elements of x which are zero.
`r`	The number of Richardson improvement iterations (repetions with successly smaller d).
`show.details`	logical indicating if detailed calculations should be shown.

Details

The first argument of a function must be a vector. span performs a svd of the tangent vectors at the point x. This can be used to calculate the dimension of the tangent space (ie. by over specifying the model and counting the number of significant singular values). This function uses Richardson extrapolation (for more details see the functions gradRichardson and genD) to get a numerical approximation of the tangent vectors to the parameter manifold. SVD is then used to calculate their span.

Value

The singular values of the matrix of tangent vectors are returned.

See Also

`span.TSestModel`, `gradRichardson`, `genD`

Examples

```
func <- function(x){c(x[1], x[1], x[2]^2)}
span(func, c(2,2))
span(func, c(2,5))
span(func, c(2,2,5))
```

`span.TSestModel` *Calculate Span of Tangent Plane*

Description

Calculate the dimension of the tangent space

Usage

```
    ## S3 method for class 'TSestModel':
    span(func, x=coef(func),
        func.args=list(Shape=TSmodel(func), data=TSdata(func)),
        d=0.01, eps=1e-4, r=6,
        show.details=FALSE, compiled=.DSEflags()$COMPILED, ...)
```

Arguments

`func`	a TSestModel object which is used as a function mapping coefficients (parameters) to residuals.
`x`	parameter vector first argument to function func indicating the point with respect to which the derivative is calculated.
`func.args`	list with additional argument to function `func`.
`d`	the fraction of x to use for the initial numerical approximation. The default is set to 0.01*x or eps if x is 0.0.
`eps`	Used instead of d for elements of x which are zero.
`r`	The number of Richardson improvement iterations (repetions with successly smaller d).

`show.details`	logical indicating if detailed calculations should be shown.
`compiled`	use the compiled version of the code. (FALSE only for debugging.)
`...`	(further arguments, currently disregarded).

Details

See the generic function.

Value

The singular values of the matrix of tangent vectors are returned.

See Also

`span`, `gradRichardson`, `genD`

Index