Package ‘Splinets’

March 6, 2023

Date 2023-02-28

Type Package

Title Functional Data Analysis using Splines and Orthogonal Spline Bases

Version 1.5.0

Author Xijia Liu [aut],
Hiba Nassar [aut],
Krzysztof Podgorski [aut, cre, cph]

Description Splines are efficiently represented through their Taylor expansion at the knots. The representation accounts for the support sets and is thus suitable for sparse functional data. Two cases of boundary conditions are considered: zero-boundary or periodic-boundary for all derivatives except the last. The periodical splines are represented graphically using polar coordinates. The B-splines and orthogonal bases of splines that reside on small total support are implemented. The orthogonal bases are referred to as 'splinets' and are utilized for functional data analysis. Random spline generator is implemented as well as all fundamental algebraic and calculus operations on splines. The optimal, in the least square sense, functional fit by 'splinets' to data consisting of sampled values of functions as well as splines build over another set of knots is obtained and used for functional data analysis. <arXiv:2102.00733>, <doi:10.1016/j.cam.2022.114444>, <arXiv:2302.07552>.

Depends R (>= 3.5.0)

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Imports methods, graphics

NeedsCompilation no

Maintainer Krzysztof Podgorski <Krzysztof.Podgorski@stat.lu.se>

Repository CRAN

Date/Publication 2023-03-06 19:40:02 UTC

R topics documented:

`construct` *Construction of a* `Splinets` *object*

Description

The function constructs a `Splinets` object correspond to a single spline (size=1) from a vector of knots and a matrix of proposed derivatives. The matrix is tested for its correctness like in `is.splinets` and adjusted using one of the implemented methods.

Usage

```
construct(knots, smorder, matder, supp = vector(), mthd = "RRM")
```

Arguments

`knots`	`n+2` vector, the knots over which the spline is built; There should be at least `2*smorder+4` of knots.
`smorder`	integer, the order of smoothness;

`matder`	`(n+2)x(smorder+1)` matrix, the matrix of derivatives; This matrix will be corrected if does not correspond to a proper spline.
`supp`	vector, either empty or two integers representing the single interval support;
`mthd`	string, one of the three methods for correction of the matrix of derivative: `'CRLC'` matching mostly the highest derivative, `'CRFC'` matching mostly the function values at the knots, `'RRM'` balanced matching between all derivatives; The default method is `'RRM'`, see the paper on the package for further details about the methods.

Details

The function constructs a `Splinet`-object only over a single interval support. Combining with the function `lincom` allows to introduce a multi-component support.

Value

A `Splinets`-object corresponding to a single spline.

References

Liu, X., Nassar, H., Podgórski, K. "Dyadic diagonalization of positive definite band matrices and efficient B-spline orthogonalization." Journal of Computational and Applied Mathematics (2022) <https://doi.org/10.1016/j.cam.2022.114444>.

Podgórski, K. (2021) "`Splinets` – splines through the Taylor expansion, their support sets and orthogonal bases." <arXiv:2102.00733>.

Nassar, H., Podgórski, K. (2023) "Splinets 1.5.0 – Periodic Splinets." <arXiv:2302.07552>

See Also

`is.splinets` for diagnostic of `Splinets`-objects; `gather` and `subsample` for combining and subsampling `Splinets`-objects, respectively, `plot,Splinets-method` for a plotting method for `Splinets`-objects; `lincom` for combining splines with more complex than a single interval support sets;

Examples

```
#--------------------------------------------------------------#
#---Building 'Splinets' using different derviative matching---#
#--------------------------------------------------------------#
n=17; k=4
set.seed(5)
xi=sort(runif(n+2)); xi[1]=0; xi[n+1]=1

#Random matrix of derivatives -- the noise (wild) case to be corrected
S=matrix(rnorm((n+2)*(k+1)),ncol=(k+1))

spl=construct(xi,k,S) #construction of an object, the order of knots is corrected
is.splinets(spl)[[1]] #validation
```

```
spl=construct(xi,k,S,mthd='CRFC') #another method of the derivative matching
is.splinets(spl)[[1]]

spl=construct(xi,k,S,mthd='CRLC') #one more method
is.splinets(spl)[[1]]

#-----------------------------------------------------#
#---------Building not over the full support----------#
#-----------------------------------------------------#
set.seed(5)
n=20; xi=sort(runif(n+2));xi[1]=0;xi[n+2]=1

spl=construct(xi,k,S) #construction of a spline as the 'Splinets' object over the entire range
is.splinets(spl)[[1]] #verification of the conditions

supp=c(3,17) #definition of the single interval support
SS=matrix(rnorm((supp[2]-supp[1]+1)*(k+1)),ncol=(k+1)) #The matrix of derivatives
                                                        #over the support range
sspl=construct(xi,k,SS,supp=supp) #construction of a spline as the 'Splinets' object
                                  #with the given support range
is.splinets(sspl)[[1]] #Verification
sspl@knots
sspl@supp
sspl@der
```

---

deriva *Derivatives of splines*

Description

The function generates a Splinets-object which contains the first order derivatives of all the splines from the input Splinets-object. The function also verifies the support set of the output to provide the accurate information about the support sets by excluding regions over which the original function is constant.

Usage

```
deriva(object, epsilon = 1e-07)
```

Arguments

object	Splinets object of the smoothness order k;
epsilon	positive number, controls removal of knots from the support; If the derivative is smaller than this number, it is considered to be zero and the corresponding knots are removed from the support.The default value is 1e-7.

Value

A `Splinets`-object of the order `k-1` that also contains the updated information about the support set.

References

Liu, X., Nassar, H., Podgórski, K. "Dyadic diagonalization of positive definite band matrices and efficient B-spline orthogonalization." Journal of Computational and Applied Mathematics (2022) <https://doi.org/10.1016/j.cam.2022.114444>.

Podgórski, K. (2021) "`Splinets` – splines through the Taylor expansion, their support sets and orthogonal bases." <arXiv:2102.00733>.

Nassar, H., Podgórski, K. (2023) "Splinets 1.5.0 – Periodic Splinets." <arXiv:2302.07552>

See Also

`integra` for generating the indefinite integral of a spline that can be viewed as the inverse operation to `deriva`; `dintegra` for the definite integral of a spline;

Examples

```
#--------------------------------------------------------#
#--- Generating the deriviative functions of splines ---#
#--------------------------------------------------------#
n=13; k=4
set.seed(5)
xi=sort(runif(n+2)); xi[1]=0; xi[n+2]=1
spl=construct(xi,k,matrix(rnorm((n+2)*(k+1)),ncol=(k+1))) #constructing three splines
spl=gather(spl, construct(xi,k,matrix(rnorm((n+2)*(k+1)),ncol=(k+1))))
spl=gather(spl, construct(xi,k,matrix(rnorm((n+2)*(k+1)),ncol=(k+1))))
# calculate the derivative of splines
dspl = deriva(spl)
plot(spl)
plot(dspl)

#----------------------------------------------#
#--- Examples with different support ranges ---#
#----------------------------------------------#

n=25; k=3
xi=seq(0,1,by=1/(n+1));
set.seed(5)
#Defining support ranges for three splines
supp=matrix(c(2,12,4,20,6,25),byrow=TRUE,ncol=2)
#Initial random matrices of the derivative for each spline
SS1=matrix(rnorm((supp[1,2]-supp[1,1]+1)*(k+1)),ncol=(k+1))
SS2=matrix(rnorm((supp[2,2]-supp[2,1]+1)*(k+1)),ncol=(k+1))
SS3=matrix(rnorm((supp[3,2]-supp[3,1]+1)*(k+1)),ncol=(k+1))
spl=construct(xi,k,SS1,supp[1,]) #constructing the first correct spline
nspl=construct(xi,k,SS2,supp[2,])
spl=gather(spl,nspl) #the second and the first ones
```

```
nspl=construct(xi,k,SS3,supp[3,])
spl=gather(spl,nspl) #the third is added

der_spl = deriva(spl)
par(mar=c(1,1,1,1))
par(mfrow=c(2,1))
plot(der_spl)
plot(spl)
par(mfrow=c(1,1))
```

---

dintegra — *Calculating the definite integral of a spline.*

Description

The function calculates the definite integrals of the splines in an input `Splinets`-object.

Usage

```
dintegra(object, sID = NULL)
```

Arguments

`object`	`Splinets`-object;
`sID`	vector of integers, the indicies specifying for which splines in the `Splinets`-object the definite integral is to be evaluated; If `sID=NULL`, then the definite integral of all splines in the object are calculated. The default is `NULL`.

Value

A `length(sID) x 2` matrix, with the first column holding the id of splines and the second column holding the corresponding definite integrals.

References

Liu, X., Nassar, H., Podgórski, K. "Dyadic diagonalization of positive definite band matrices and efficient B-spline orthogonalization." Journal of Computational and Applied Mathematics (2022) <https://doi.org/10.1016/j.cam.2022.114444>.

Podgórski, K. (2021) "`Splinets` – splines through the Taylor expansion, their support sets and orthogonal bases." <arXiv:2102.00733>.

Nassar, H., Podgórski, K. (2023) "Splinets 1.5.0 – Periodic Splinets." <arXiv:2302.07552>

See Also

`integra` for generating the indefinite integral; `deriva` for generating derivative functions of splines;

Examples

```
#------------------------------------------#
#--- Example with common support ranges ---#
#------------------------------------------#
n=23; k=4
set.seed(5)
xi=sort(runif(n+2)); xi[1]=0; xi[n+2]=1
# generate a random matrix S
S=matrix(rnorm((n+2)*(k+1)),ncol=(k+1))
# construct the spline
spl=construct(xi,k,S) #constructing the first correct spline
spl=gather(spl,construct(xi,k,S,mthd='CRFC')) #the second and the first ones
spl=gather(spl,construct(xi,k,S,mthd='CRLC')) #the third is added

plot(spl)
dintegra(spl, sID = c(1,3))
dintegra(spl)
plot(spl,sID=c(1,3))

#---------------------------------------------#
#--- Examples with different support ranges---#
#---------------------------------------------#

n=25; k=2
xi=seq(0,1,by=1/(n+1))
#Defining support ranges for three splines
supp=matrix(c(2,12,4,20,6,25),byrow=TRUE,ncol=2)
#Initial random matrices of the derivative for each spline
set.seed(5)
SS1=matrix(rnorm((supp[1,2]-supp[1,1]+1)*(k+1)),ncol=(k+1))
SS2=matrix(rnorm((supp[2,2]-supp[2,1]+1)*(k+1)),ncol=(k+1))
SS3=matrix(rnorm((supp[3,2]-supp[3,1]+1)*(k+1)),ncol=(k+1))
spl=construct(xi,k,SS1,supp[1,]) #constructing the first correct spline
nspl=construct(xi,k,SS2,supp[2,])
spl=gather(spl,nspl) #the second and the first ones
nspl=construct(xi,k,SS3,supp[3,])
spl=gather(spl,nspl) #the third is added

plot(spl)
dintegra(spl, sID = 1)
dintegra(spl)

#The third order case
n=40; xi=seq(0,1,by=1/(n+1)); k=3;
support=list(matrix(c(2,12,15,27,30,40),ncol=2,byrow = TRUE))
sp=new("Splinets",knots=xi,smorder=k,supp=support)
m=sum(sp@supp[[1]][,2]-sp@supp[[1]][,1]+1) #the number of knots in the support
sp@der=list(matrix(rnorm(m*(k+1)),ncol=(k+1))); sp1 = is.splinets(sp)[[2]]

support=list(matrix(c(2,13,17,30),ncol=2,byrow = TRUE))
sp=new("Splinets",knots=xi,smorder=k,supp=support)
m=sum(sp@supp[[1]][,2]-sp@supp[[1]][,1]+1) #the number of knots in the support
```

```
sp@der=list(matrix(rnorm(m*(k+1)),ncol=(k+1))); sp2 = is.splinets(sp)[[2]]

sp = gather(sp1,sp2)
dintegra(sp)
plot(sp)

lcsp=lincomb(sp,matrix(c(-1,1),ncol=2))
dintegra(lcsp)               #linearity of the integral
dintegra(sp2)-dintegra(sp1)
```

---

evspline *Evaluating splines at given arguments.*

Description

For a Splinets-object S and a vector of arguments t, the function returns the matrix of values for the splines in S. The evaluations are done through the Taylor expansions, so on the ith interval for $t \in [\xi_i, \xi_{i+1}]$:

$$S(t) = \sum_{j=0}^{k} s_{ij} \frac{(t-\xi_i)^j}{j!}.$$

For the zero order splines which are discontinuous at the knots, the following convention is taken. At the LHS knots the value is taken as the RHS-limit, at the RHS knots as the LHS-limit. The value at the central knot for the zero order and an odd number of knots case is assumed to be zero.

Usage

```
evspline(object, sID = NULL, x = NULL, N = 250)
```

Arguments

object	Splinets object;
sID	vector of integers, the indicies specifying splines in the Splinets list to be evaluated; If sID=NULL, then all splines in the Splinet-object are evaluated. The default value is NULL.
x	vector, the arguments at which the splines are evaluated; If x is NULL, then the splines are evaluated over regular grids per each interval of the support. The default value is x=NULL.
N	integer, the number of points per an interval between two consequitive knots at which the splines are evaluated. The default value is N = 250;

Value

The length(x) x length(sID+1) matrix containing the argument values, in the first column, then, columnwise, values of the subsequent splines.

References

Liu, X., Nassar, H., Podgórski, K. "Dyadic diagonalization of positive definite band matrices and efficient B-spline orthogonalization." Journal of Computational and Applied Mathematics (2022) <https://doi.org/10.1016/j.cam.2022.114444>.

Podgórski, K. (2021) "`Splinets` – splines through the Taylor expansion, their support sets and orthogonal bases." <arXiv:2102.00733>.

Nassar, H., Podgórski, K. (2023) "Splinets 1.5.0 – Periodic Splinets." <arXiv:2302.07552>

See Also

`is.splinets` for diagnostic of `Splinets`-objects; `plot,Splinets-method` for plotting `Splinets`-objects;

Examples

```
#--------------------------------------------#
#-- Example piecewise polynomial vs. spline --#
#--------------------------------------------#
n=20; k=3; xi=sort(runif(n+2))
sp=new("Splinets",knots=xi)

#Randomly assigning the derivatives -- a very 'wild' function.
S=matrix(rnorm((n+2)*(k+1)),ncol=(k+1))
sp@supp=list(t(c(1,n+2))); sp@smorder=k; sp@der[[1]]=S

y = evspline(sp)
plot(y,type = 'l',col='red')

#A correct spline object
nsp=is.splinets(sp)
sp2=nsp$robject
y = evspline(sp2)
lines(y,type='l')

#--------------------------------------------#
#-- Example piecewise polynomial vs. spline --#
#--------------------------------------------#
#Gathering three 'Splinets' objects using three different
#method to correct the derivative matrix
n=17; k=4; xi=sort(runif(n+2)); xi[1]=0; xi[n+2]=1

S=matrix(rnorm((n+2)*(k+1)),ncol=(k+1)) # generate a random matrix S

spl=construct(xi,k,S) #constructing the first correct spline
spl=gather(spl,construct(xi,k,S,mthd='CRFC')) #the second and the first ones
spl=gather(spl,construct(xi,k,S,mthd='CRLC')) #the third is added
y = evspline(spl, sID= 1)
plot(y,type = 'l',col='red')

y = evspline(spl, sID = c(1,3))
plot(y[,1:2],type = 'l',col='red')
```

```
points(y[,c(1,3)],type = 'l',col='blue')

#sID = NULL
y = evspline(spl)
plot(y[,1:2],type = 'l',col='red',ylim=range(y[,2:4]))
points(y[,c(1,3)],type = 'l',col='blue')
points(y[,c(1,4)],type = 'l',col='green')

#--------------------------------------------#
#--- Example with different support ranges ---#
#--------------------------------------------#
n=25; k=3; xi=sort(runif(n+2)); xi[1]=0; xi[n+2]=1
#Defining support ranges for three splines
supp=matrix(c(2,12,4,20,6,25),byrow=TRUE,ncol=2)
#Initial random matrices of the derivative for each spline
SS1=matrix(rnorm((supp[1,2]-supp[1,1]+1)*(k+1)),ncol=(k+1))
SS2=matrix(rnorm((supp[2,2]-supp[2,1]+1)*(k+1)),ncol=(k+1))
SS3=matrix(rnorm((supp[3,2]-supp[3,1]+1)*(k+1)),ncol=(k+1))
spl=construct(xi,k,SS1,supp[1,]) #constructing the first correct spline
nspl=construct(xi,k,SS2,supp[2,],'CRFC')
spl=gather(spl,nspl) #the second and the first ones
nspl=construct(xi,k,SS3,supp[3,],'CRLC')
spl=gather(spl,nspl) #the third is added

y = evspline(spl, sID= 1)
plot(y,type = 'l',col='red')

y = evspline(spl, sID = c(1,3))
plot(y[,1:2],type = 'l',col='red')
points(y[,c(1,3)],type = 'l',col='blue')

#sID = NULL -- all splines evaluated
y = evspline(spl)
plot(y[,c(1,3)],type = 'l',col='red',ylim=c(-1,1))
points(y[,1:2],type = 'l',col='blue')
points(y[,c(1,4)],type = 'l',col='green')
```

---

exsupp — *Correcting support sets and reshaping the matrix of derivatives at the knots.*

Description

The function is adjusting for a potential reduction in the support sets due to negligibly small values of rows in the derivative matrix. If the derivative matrix has a row equal to zero (or smaller than a neglible positive value) in the one-sided representation of it (see the references and sym2one), then the corresponding knot should be removed from the support set. The function can be used to eliminate the neglible support components from a `Splinets`-object.

Usage

```
exsupp(S, supp = NULL, epsilon = 1e-07)
```

Arguments

S	(m+2)x(k+1) matrix, the values of the derivatives at the knots over some input support set which has the cardinality m+2; The matrix is assumed to be in the symmetric around center form for each component of the support.
supp	NULL or Nsupp x2 matrix of integers, the endpoints indices for the input support intervals, where Nsupp is the number of the components in the support set; If the parameter is NULL, than the full support is assumed.
epsilon	small positive number, threshold value of the norm of rows of S; If the norm of a row of S is less than epsilon, then it will be viewed as a neglible and the knot is excluded from the inside of the support set.

Details

This function typically would be applied to an element in the list given by SLOT der of a Splinets-object. It eliminates from the support sets regions of negligible values of a corresponding spline and its derivatives.

Value

The list of two elements: exsupp$rS is the reduced derivative matrix from which the neglible rows, if any, have been removed and exsupp$rsupp is the corresponding reduced support. The output matrix has all the support components in the symmetric around the center form, which is how the derivatives are kept in the Splinets-objects.

References

Liu, X., Nassar, H., Podgórski, K. "Dyadic diagonalization of positive definite band matrices and efficient B-spline orthogonalization." Journal of Computational and Applied Mathematics (2022) <https://doi.org/10.1016/j.cam.2022.114444>.

Podgórski, K. (2021) "Splinets – splines through the Taylor expansion, their support sets and orthogonal bases." <arXiv:2102.00733>.

Nassar, H., Podgórski, K. (2023) "Splinets 1.5.0 – Periodic Splinets." <arXiv:2302.07552>

See Also

Splinets-class for the description of the Splinets-class; sym2one for switching between the representations of a derivative matrix over a general support set; lincomb for evaluating a linear transformation of splines in a Splinets-object; is.splinets for a diagnostic tool of the Splinets-objects;

Examples

```
#----------------------------------------------------#
#---Correcting support sets in a derivative matrix---#
#----------------------------------------------------#
n=20; k=3; xi=seq(0,1,by=1/(n+1)) #an even number of equally spaced knots
set.seed(5)
S=matrix(rnorm((n+2)*(k+1)),ncol=(k+1))
spl=construct(xi,k,S) #this spline will be used below to construct a 'sparse' spline
is.splinets(spl) #verification
plot(spl)


xxi=seq(0,20,by=1/(n+1)) #large set of knots for construction of a sparse spline
nn=length(xxi)-2
spspl=new('Splinets',knots=xxi,smorder=k) #generic object from the 'Splinets'-class
spspl@der[[1]]=matrix(0,ncol=(k+1),nrow=(nn+2)) #starting with zeros everywhere

spspl@der[[1]][1:(n+2),]=sym2one(spl@der[[1]]) #assigning local spline to a sparse spline at
spspl@der[[1]][nn+3-(1:(n+2)),]=spspl@der[[1]][(n+2):1,] #the beginning and the same at the end
spspl@der[[1]]=sym2one(spspl@der[[1]],inv=TRUE)
                                 #at this point the object does not account for the sparsity

is.splinets(spspl) #a sparse spline on 421 knots with a non-zero terms at the first 22
                     #and at the last 22 knots, the actual support set is not yet reported
plot(spspl)
plot(spspl,xlim=c(0,1)) #the local part of the sparse spline

exsupp(spspl@der[[1]]) #the actual support of the spline given the sparse derivative matrix

#Expanding the previous spline by building a slightly more complex support set
spspl@der[[1]][(n+1)+(1:(n+2)),]=sym2one(spl@der[[1]]) #double the first component of the
                                                  #support because these are tangent supports
spspl@der[[1]][(2*n+3)+(1:(n+2)),]=sym2one(spl@der[[1]]) #tdetect a single component of
                                                   #the support with no internal knots removed
is.splinets(spspl)
plot(spspl)


es=exsupp(spspl@der[[1]])
es[[2]]   #the new support made of three components with the two first ones
          #separated by an interval with no knots in it


spspl@der[[1]]=es[[1]]         #defining the spline on the evaluated actual support
spspl@supp[[1]]=es[[2]]
#Example with reduction of not a full support.

xi1=seq(0,14/(n+1),by=1/(n+1)); n1=13; #the odd number of equally spaced knots
S1=matrix(rnorm((n1+2)*(k+1)),ncol=(k+1))
spl1=construct(xi1,k,S1) #construction of a local spline
xi2=seq(16/(n+1),42/(n+1),by=1/(n+1)); n2=25; #the odd number of equally spaced knots
```

```
S2=matrix(rnorm((n2+2)*(k+1)),ncol=(k+1))
spl2=construct(xi2,k,S2) #construction of a local spline

spspl@der[[1]][1:15,]=sym2one(spl1@der[[1]])
spspl@der[[1]][16,]=rep(0,k+1)
spspl@der[[1]][17:43,]=sym2one(spl2@der[[1]])
spspl@der[[1]][1:43,]=sym2one(spspl@der[[1]][1:43,],inv=TRUE)

is.splinets(spspl) #three intervals in the support are repported

exsupp(spspl@der[[1]],spspl@supp[[1]])
```

`gather` *Combining two* `Splinets` *objects*

Description

The function returns the `Splinets`-object that gathers two input `Splinets`-objects together. The input objects have to be of the same order and over the same knots.

Usage

```
gather(Sp1, Sp2)
```

Arguments

`Sp1`	`Splinets` object;
`Sp2`	`Splinets` object;

Value

`Splinets` object, contains grouped splines from the input objects;

References

Liu, X., Nassar, H., Podgórski, K. "Dyadic diagonalization of positive definite band matrices and efficient B-spline orthogonalization." Journal of Computational and Applied Mathematics (2022) <https://doi.org/10.1016/j.cam.2022.114444>.

Podgórski, K. (2021) "`Splinets` – splines through the Taylor expansion, their support sets and orthogonal bases." <arXiv:2102.00733>.

Nassar, H., Podgórski, K. (2023) "Splinets 1.5.0 – Periodic Splinets." <arXiv:2302.07552>

See Also

`is.splinets` for diagnostic of the `Splinets`-objects; `construct` for constructing such an object; `subsample` for subsampling `Splinets`-objects; `plot,Splinets-method` for plotting `Splinets`-objects;

Examples

```
#-------------------------------------------------------------#
#-----------------Grouping into a 'Splinets' object-----------#
#-------------------------------------------------------------#
#Gathering three 'Splinets' objects using three different
#method to correct the derivative matrix
set.seed(5)
n=13;xi=sort(runif(n+2)); xi[1]=0; xi[n+2]=1; k=4
S=matrix(rnorm((n+2)*(k+1)),ncol=(k+1))

spl=construct(xi,k,S) #constructing the first correct spline
spl=gather(spl,construct(xi,k,S,mthd='CRFC')) #the second and the first ones
spl=gather(spl,construct(xi,k,S,mthd='CRLC')) #the third is added

is.splinets(spl)[[1]] #diagnostic
spl@supp #the entire range for the support

#Example with different support ranges, the 3rd order
set.seed(5)
n=25; xi=sort(runif(n+2)); xi[1]=0; xi[n+2]=1; k=3
supp=list(t(c(2,12)),t(c(4,20)),t(c(6,25))) #support ranges for three splines

#Initial random matrices of the derivative for each spline
SS1=matrix(rnorm((supp[[1]][1,2]-supp[[1]][1,1]+1)*(k+1)),ncol=(k+1))
SS2=matrix(rnorm((supp[[2]][1,2]-supp[[2]][1,1]+1)*(k+1)),ncol=(k+1))
SS3=matrix(rnorm((supp[[3]][1,2]-supp[[3]][1,1]+1)*(k+1)),ncol=(k+1))

spl=construct(xi,k,SS1,supp[[1]]) #constructing the first correct spline
nspl=construct(xi,k,SS2,supp[[2]])
spl=gather(spl,nspl) #the second and the first ones
nspl=construct(xi,k,SS3,supp[[3]])
spl=gather(spl,nspl) #the third is added
is.splinets(spl)[[1]]

spl@supp
spl@der
```

---

gramian *Gramian matrix, norms, and inner products of splines*

Description

The function performs evaluation of the matrix of the inner products $\int S(t) \cdot T(t)dt$ of all the pairs of splines S, T from the input object. The program utilizes the Taylor expansion of splines, see the reference for details.

Usage

```
gramian(Sp, norm_only = FALSE, sID = NULL, Sp2 = NULL, s2ID = NULL)
```

Arguments

`Sp`	`Splinets` object;
`norm_only`	logical, indicates if only the square norm of the elements in the input object is calculated; The default is `norm_only=FALSE`;
`sID`	vector of integers, the indicies specifying splines in the `Splinets` list `Sp` to be evaluated; If `sID=NULL` (default), then the inner products for all the pairs taken from the object are evaluated.
`Sp2`	`Splinets` object, the optional second `Splinets`-object; The inner products between splines in `Sp` and in `Sp2` are evaluated, i.e. the cross-gramian matrix.
`s2ID`	vector of integers, the indicies specifying splines in the `Sp2` to be considered in the cross-gramian;

Details

If there is only one input `Splinet`-object, then the non-negative symmetrix matrix of the splines in this object is returned. If there are two input `Splinet`-objects, then the $m \times r$ matrix of the cross-inner product is returned, where m is the number of splines in the first object and r is their number in the second one. If only the norms are evaluated (`norm_only= TRUE`) it is always evaluating the norms of the first object. In the case of two input `Splinets`-objects, they should be over the same set of knots and of the same smoothness order.

Value

- `norm_only=FALSE` – the Gram matrix of inner products of the splines within the input `Splinets`-objects is returned,
- `Sp2 = NULL` – the non-negative definite matrix of the inner products of splines in `Sp` is returned,
- both `Sp` and `Sp2` are non-`NULL` and contain splines S_i's and T_j's, respectively == the cross-gramian matris of the inner products for the pairs of splines (S_i, T_j) is returned,
- `norm_only=FALSE`– the vector of the norms of `Sp` is returned.

References

Liu, X., Nassar, H., Podgórski, K. "Dyadic diagonalization of positive definite band matrices and efficient B-spline orthogonalization." Journal of Computational and Applied Mathematics (2022) <https://doi.org/10.1016/j.cam.2022.114444>.

Podgórski, K. (2021) "`Splinets` – splines through the Taylor expansion, their support sets and orthogonal bases." <arXiv:2102.00733>.

Nassar, H., Podgórski, K. (2023) "Splinets 1.5.0 – Periodic Splinets." <arXiv:2302.07552>

See Also

`lincomb` for evaluation of a linear combination of splines; `project` for projections to the spaces of Splines;

Examples

```
#--------------------------------------#
#---- Simple three splines example -----#
#--------------------------------------#
n=25; k=3
xi=sort(runif(n+2)); xi[1]=0; xi[n+2]=1
#Defining support ranges for three splines
supp=matrix(c(2,12,4,20,6,25),byrow=TRUE,ncol=2)
#Initial random matrices of the derivative for each spline
SS1=matrix(rnorm((supp[1,2]-supp[1,1]+1)*(k+1)),ncol=(k+1))
SS2=matrix(rnorm((supp[2,2]-supp[2,1]+1)*(k+1)),ncol=(k+1))
SS3=matrix(rnorm((supp[3,2]-supp[3,1]+1)*(k+1)),ncol=(k+1))
spl=construct(xi,k,SS1,supp[1,]) #constructing the first correct spline
nspl=construct(xi,k,SS2,supp[2,])
spl=gather(spl,nspl) #the second and the first ones
nspl=construct(xi,k,SS3,supp[3,])
spl=gather(spl,nspl) #the third is added

plot(spl)
gramian(spl)
gramian(spl, norm_only = TRUE)
gramian(spl, sID = c(1,3))
gramian(spl,sID=c(2,3),Sp2=spl,s2ID=c(1)) #the cross-Gramian matrix

#-----------------------------------------#
#--- Example with varying support sets ---#
#-----------------------------------------#
n=40; xi=seq(0,1,by=1/(n+1)); k=2;
support=list(matrix(c(2,9,15,24,30,37),ncol=2,byrow = TRUE))
sp=new("Splinets",knots=xi,smorder=k,supp=support)
m=sum(sp@supp[[1]][,2]-sp@supp[[1]][,1]+1) #the number of knots in the support
sp@der=list(matrix(rnorm(m*(k+1)),ncol=(k+1))) #the derivative matrix at random
sp1 = is.splinets(sp)[[2]] #the correction of 'der' matrices

support=list(matrix(c(5,12,17,29),ncol=2,byrow = TRUE))
sp=new("Splinets",knots=xi,smorder=k,supp=support)
m=sum(sp@supp[[1]][,2]-sp@supp[[1]][,1]+1) #the number of knots in the support
sp@der=list(matrix(rnorm(m*(k+1)),ncol=(k+1))) #the derivative matrix at random
sp2 = is.splinets(sp)[[2]]

spp = gather(sp1,sp2)

support=list(matrix(c(3,10,14,21,27,34),ncol=2,byrow = TRUE))
sp=new("Splinets",knots=xi,smorder=k,supp=support)
m=sum(sp@supp[[1]][,2]-sp@supp[[1]][,1]+1) #the number of knots in the support
sp@der=list(matrix(rnorm(m*(k+1)),ncol=(k+1))) #the derivative matrix at random
sp3 = is.splinets(sp)[[2]]

spp = gather(spp, sp3)

plot(spp)
gramian(spp) #the regular gramian matrix
```

```
spp2=subsample(spp,sample(1:3,size=3,rep=TRUE))
gramian(Sp=spp,Sp2=spp2) #cross-Gramian matrix


#----------------------------------------#
#--------- Grammian for B-splines --------#
#----------------------------------------#

n=25; xi=seq(0,1,by=1/(n+1)); k=2;
Sp=splinet(xi) #B-splines and corresponding splinet

gramian(Sp$bs) #band grammian matrix for B-splines
gramian(Sp$os) #diagonal gramian matrix for the splinet
A=gramian(Sp=Sp$bs,Sp2=Sp$os) #cross-Gramian matrix, the coefficients of
                              #the decomposition of the B-splines

plot(Sp$bs)
plot(lincomb(Sp$os,A))
```

`integra` *Indefinite integrals of splines*

Description

The function generates the indefinite integrals for given input splines. The integral is a function of the upper limit of the definite integral and is a spline of the higher order that does not satisfy the zero boundary conditions at the RHS endpoint, unless the definite integral over the whole range is equal to zero. Moreover, the support of the function is extended in the RHS up to the RHS end point unless the definite integral of the input is zero, in which the case the support is extracted from the obtained spline.

Usage

```
integra(object, epsilon = 1e-07)
```

Arguments

`object`	a `Splinets` object of the smoothness order `k`;
`epsilon`	non-negative number indicating accuracy when close to zero value are detected; This accuracy is used in when the boundary conditions of the integral are checked.

Details

The value on the RHS is not zero, so the zero boundary condition typically is not satisfied and the support is is extended to the RHS end of the whole domain of splines. However, the function returns proper support if the original spline is a derivative of a spline that satisfies the boundary conditons.

Value

A `Splinets`-object with order `k+1` that contains the indefinite integrals of the input object.

References

Liu, X., Nassar, H., Podgórski, K. "Dyadic diagonalization of positive definite band matrices and efficient B-spline orthogonalization." Journal of Computational and Applied Mathematics (2022) <https://doi.org/10.1016/j.cam.2022.114444>.

Podgórski, K. (2021) "`Splinets` – splines through the Taylor expansion, their support sets and orthogonal bases." <arXiv:2102.00733>.

Nassar, H., Podgórski, K. (2023) "Splinets 1.5.0 – Periodic Splinets." <arXiv:2302.07552>

See Also

`deriva` for computing derivatives of splines; `dintegra` for the definite integral;

Examples

```
#-------------------------------------#
#--- Generate indefinite integral ---#
#-------------------------------------#
n=18; k=3; xi=sort(runif(n+2)); xi[1]=0; xi[n+2]=1
# generate a random matrix S
set.seed(5)
S=matrix(rnorm((n+2)*(k+1)),ncol=(k+1))
spl=construct(xi,k,S) #constructing a spline
plot(spl)

dspl = deriva(spl) #derivative
plot(dspl)
is.splinets(dspl)
dintegra(dspl) #the definite integral is 0 (the boundary conditions for 'spl')

ispl = integra(spl) #the integral of a spline
plot(ispl) #the boundary condition on the rhs not satisfied (non-zero value)
ispl@smorder
is.splinets(ispl) #the object does not satisfy the boundary condition for the spline

spll = integra(dspl)
plot(spll)
is.splinets(spll) #the boundary conditions of the integral of the derivative satisfied.


#----------------------------------------------#
#--- Examples with different support ranges ---#
#----------------------------------------------#
n=25; k=2;
set.seed(5)
xi=sort(runif(n+2)); xi[1]=0; xi[n+2]=1
#Defining support ranges for three splines
supp=matrix(c(2,12,4,20,6,25),byrow=TRUE,ncol=2)
#Initial random matrices of the derivative for each spline
SS1=matrix(rnorm((supp[1,2]-supp[1,1]+1)*(k+1)),ncol=(k+1))
SS2=matrix(rnorm((supp[2,2]-supp[2,1]+1)*(k+1)),ncol=(k+1))
SS3=matrix(rnorm((supp[3,2]-supp[3,1]+1)*(k+1)),ncol=(k+1))
```

```
spl=construct(xi,k,SS1,supp[1,]) #constructing the first proper spline
nspl=construct(xi,k,SS2,supp[2,],'CRFC')
spl=gather(spl,nspl) #the second and the first together
nspl=construct(xi,k,SS3,supp[3,],'CRLC')
spl=gather(spl,nspl) #the third is added

plot(spl)
spl@supp


dspl = deriva(spl) #derivative of the splines
plot(dspl)

dintegra(dspl) #the definite integral over the entire range of knots is zero

idspl = integra(dspl) #integral of the derivative returns the original splines
plot(idspl)
is.splinets(idspl)  #and confirms that the object is a spline with boundary conditions
                    #satified


idspl@supp #Since integral is taken over a function that integrates to zero over
spl@supp  #each of the support interval, the support of all three objects are the same.
dspl@supp

ispl=integra(spl)
plot(ispl) #the zero boundary condition at the RHS-end for the splines are not satisfied.
is.splinets(ispl) #thus the object is reported as a non-spline

plot(deriva(ispl))
displ=deriva(ispl)
displ@supp #Comparison of the supports
spl@supp
           #Here the integrals have extended support as it is taken from a function
ispl@supp  #that does not integrate to zero.



#---------------------------------------#
#---Example with complicated supports---#
#---------------------------------------#
n=40; xi=seq(0,1,by=1/(n+1)); k=3;
support=list(matrix(c(2,12,15,27,30,40),ncol=2,byrow = TRUE))
sp=new("Splinets",knots=xi,smorder=k,supp=support)
m=sum(sp@supp[[1]][,2]-sp@supp[[1]][,1]+1) #the number of knots in the support
sp@der=list(matrix(rnorm(m*(k+1)),ncol=(k+1))) #the derivative matrix at random
sp1 = is.splinets(sp)[[2]] #Comparison of the corrected and the original 'der' matrices

support=list(matrix(c(2,13,17,30),ncol=2,byrow = TRUE))
sp=new("Splinets",knots=xi,smorder=k,supp=support)
m=sum(sp@supp[[1]][,2]-sp@supp[[1]][,1]+1) #the number of knots in the support
sp@der=list(matrix(rnorm(m*(k+1)),ncol=(k+1))) #the derivative matrix at random
sp2 = is.splinets(sp)[[2]]
```

```
sp = gather(sp1,sp2) #a group of two splines
plot(sp)

dsp = deriva(sp) #derivative
plot(dsp)

spl = integra(dsp)
plot(spl) #the spline retrieved
spl@supp  #the supports are retrieved as well
sp@supp
is.splinets(spl) #the proper splinet object that satisfies the boundaries

ispl = integra(sp)
plot(ispl)
ispl@supp #full support shown by empty list in SLOT 'supp'
is.splinets(ispl) #diagnostic confirms no zeros at the boundaries

spll = deriva(ispl)
plot(spll)
spll@supp
```

`is.splinets` *Diagnostics of splines and their generic correction*

Description

The method performs verification of the properties of SLOTS of an object belonging to the `Splinets`–class. In the case when all the properties are satisfied the logical `TRUE` is returned. Otherwise, `FALSE` is returned together with suggested corrections.

Usage

```
is.splinets(object)
```

Arguments

`object`	`Splinets` object, the object to be diagnosed; For this object to be corrected properly each support interval has to have at least `2*smorder+4` knots.

Value

A list made of: a logical value `is`, a `Splinets` object `robject`, and a numeric value `Er`.

- The logical value `is` indicates if all the condtions for the elements of `Splinets` object to be a collection of valid splines are satisfied, additional diagnostic messages are printed out.
- The object `robject` is a modified input object that has all SLOT fields modified so the conditions/restrictions to be a proper spline are satisfied.
- The numeric value `Er` is giving the total squared error of deviation of the input matrix of derivative from the conditions required for a spline.

References

Liu, X., Nassar, H., Podgórski, K. "Dyadic diagonalization of positive definite band matrices and efficient B-spline orthogonalization." Journal of Computational and Applied Mathematics (2022) <https://doi.org/10.1016/j.cam.2022.114444>.

Podgórski, K. (2021) "Splinets – splines through the Taylor expansion, their support sets and orthogonal bases." <arXiv:2102.00733>.

Nassar, H., Podgórski, K. (2023) "Splinets 1.5.0 – Periodic Splinets." <arXiv:2302.07552>

See Also

`Splinets-class` for the definition of the `Splinets`-class; `construct` for constructing such an object from the class; `gather` and `subsample` for combining and subsampling `Splinets`-objects, respectively; `plot,Splinets-method` for plotting `Splinets` objects;

Examples

```
#-----------------------------------------------------#
#--------Diagnostics of simple Splinets objects-------#
#-----------------------------------------------------#
#----------Full support equidistant cases-------------#
#-----------------------------------------------------#
#Zero order splines, equidistant case, full support
n=20; xi=seq(0,1,by=1/(n+1))
sp=new("Splinets",knots=xi)
sp@equid #equidistance flag
#Diagnostic of 'Splinets' object 'sp'
is.splinets(sp)

IS=is.splinets(sp)
IS[[1]] #informs if the object is a spline
IS$is   #equivalent to the above

#Third order splines with a noisy matrix of the derivative
set.seed(5)
k=3; sp@smorder=k; sp@der[[1]]=matrix(rnorm((n+2)*(k+1)),ncol=(k+1))

IS=is.splinets(sp)
IS[[2]]@taylor #corrections
sp@taylor
IS[[2]]@der #corrections
sp@der

is.splinets(IS[[2]]) #The output object is a valid splinet

#-----------------------------------------------------#
#--------Full support non-equidistant cases-----------#
#-----------------------------------------------------#
#Zero order splines, non-equidistant case, full support
set.seed(5)
n=17; xi=sort(runif(n+2))
xi[1]=0 ;xi[n+1]=1 #The last knot is not in the order.
```

```
          #It will be reported and corrected in the output.
sp=new("Splinets",knots=xi)

xi #original knots
sp@knots #vs. corrected ones
sp@taylor

#Diagnostic of 'Splinets' object 'sp'
is.splinets(sp)

IS=is.splinets(sp)

nsp=IS$robject #the output spline -- a corrected version of the input
nsp@der
sp@der

#Third order splines
nsp@smorder=3
IS=is.splinets(nsp)

IS[[2]]@taylor #corrections
nsp@taylor
IS[[2]]@der #corrections
nsp@der
is.splinets(IS[[2]]) #verification that the correction is a valid object

#Randomly assigning the derivative -- a very 'unstable' function.
set.seed(5)
k=nsp@smorder; S=matrix(rnorm((n+2)*(k+1)),ncol=(k+1)); nsp@der[[1]]=S

IS=is.splinets(nsp) #the 2nd element of 'IS' is a spline obtained by correcting 'S'
nsp=is.splinets(IS[[2]])
nsp$is #The 'Splinets' object is correct, alternatively use 'nsp$[[1]]'.
nsp$robject #A correct spline object, alternatively use 'nsp$[[2]]'.

#-----------------------------------------------------#
#-----Splinets objects with varying support sets------#
#-----------------------------------------------------#
#-----------------Eequidistant cases------------------#
#-----------------------------------------------------#
#Zero order splines, equidistant case, support with three components
n=20; xi=seq(0,1,by=1/(n+1))
support=list(matrix(c(2,5,6,8,12,18),ncol=2,byrow = TRUE))
sp=new("Splinets",knots=xi,supp=support)
is.splinets(sp)

IS=is.splinets(sp)

sum(sp@supp[[1]][,2]-sp@supp[[1]][,1]+1) #the number of knots in the support
dim(IS[[2]]@der[[1]])[1]  #the number of rows in the derivative matrix
IS[[2]]@der[[1]] #the corrected object
sp@der #the input derivative matrix
```

```
#Third order splines
n=40; xi=seq(0,1,by=1/(n+1)); k=3;
support=list(matrix(c(2,12,15,27,30,40),ncol=2,byrow = TRUE))
m=sum(support[[1]][,2]-support[[1]][,1]+1) #the number of knots in the support
SS=list(matrix(rnorm(m*(k+1)),ncol=(k+1))) #the derivative matrix at random

sp=new("Splinets",knots=xi,smorder=k,supp=support,der=SS)

IS=is.splinets(sp)

m=sum(sp@supp[[1]][,2]-sp@supp[[1]][,1]+1) #the number of knots in the support
sp@der=list(matrix(rnorm(m*(k+1)),ncol=(k+1))) #the derivative matrix at random

IS=is.splinets(sp) #Comparison of the corrected and the original 'der' matrices
sp@der
IS[[2]]@der
is.splinets(IS[[2]]) #verification

#-----------------------------------------------------#
#----------------Non-equidistant cases----------------#
#-----------------------------------------------------#
#Zero order splines, non-equidistant case, support with three components
set.seed(5)
n=43; xi=seq(0,1,by=1/(n+1)); k=3; xi=sort(runif(n+2)); xi[1]=0; xi[n+2]=1;
support=list(matrix(c(2,14,17,30,32,43),ncol=2,byrow = TRUE))

ssp=new("Splinets",knots=xi,supp=support) #with partial support
nssp=is.splinets(ssp)$robject
nssp@supp
nssp@der

#Third order splines
nssp@smorder=3 #changing the order of the 'Splinets' object
set.seed(5)
m=sum(nssp@supp[[1]][,2]-nssp@supp[[1]][,1]+1) #the number of knots in the support
nssp@der=list(matrix(rnorm(m*(k+1)),ncol=(k+1)))  #the derivative matrix at random
IS=is.splinets(nssp)
IS$robject@der
is.splinets(IS$robject)$is #verification of the corrected output object
```

---

is.splinets,Splinets-method
Diagnostics of splines

Description

This short information is added to satisfy an R-package building requirement, see `is.splinets` for the actual information.

Usage

```
## S4 method for signature 'Splinets'
is.splinets(object)
```

Arguments

`object`	`Splinets` object, the object to be diagnosed;

`lincomb` *Linear transformation of splines.*

Description

A linear combination of the splines S_j in the input object is computed according to

$$R_i = \sum_{j=0}^{d} a_{ij} S_j,\ i = 1, \ldots, l.$$

and returned as a `Splinet`-object.

Usage

```
lincomb(object, A, reduced = TRUE, SuppExtr = TRUE)
```

Arguments

`object`	`Splinets` object containing `d` splines;
`A`	`l x d` matrix; coefficients of the linear transformation,
`reduced`	logical; If `TRUE` (default), then the linear combination is calculated accounting for the actual support sets (recommended for sparse splines), if `FALSE`, then the full support computations are used (can be faster for lower dimension or non-sparse cases).
`SuppExtr`	logical; If `TRUE` (default), the true support is extracted, otherwise, full range is reported as the support. Applies only to the case when `reduced=FALSE`.

Value

A `Splinet`-object that contains `l` splines obtained by linear combinations of using coefficients in rows of `A`. The `SLOT` `type` of the output splinet objects is `sp`.

References

Liu, X., Nassar, H., Podgórski, K. "Dyadic diagonalization of positive definite band matrices and efficient B-spline orthogonalization." Journal of Computational and Applied Mathematics (2022) <https://doi.org/10.1016/j.cam.2022.114444>.

Podgórski, K. (2021) "`Splinets` – splines through the Taylor expansion, their support sets and orthogonal bases." <arXiv:2102.00733>.

Nassar, H., Podgórski, K. (2023) "Splinets 1.5.0 – Periodic Splinets." <arXiv:2302.07552>

See Also

exsupp for extracting the correct support; construct for building a valid spline; rspline for random generation of splines;

Examples

```
#-------------------------------------------------------------#
#------------Simple linear operations on Splinets-------------#
#-------------------------------------------------------------#
#Gathering three 'Splinets' objects
n=53; k=4; xi=sort(runif(n+2)); xi[1]=0; xi[n+2]=1;Nspl=10
set.seed(5)
S=matrix(rnorm((n+2)*(k+1)),ncol=(k+1))
spl=construct(xi,k,S) #constructing the first proper spline
spl@epsilon=1.0e-5 #to avoid FALSE in the next function due to inaccuracies
is.splinets(spl)

RS=rspline(spl,Nspl) #Random splines
plot(RS)
A = matrix(rnorm(5*Nspl, mean = 2, sd = 100), ncol = Nspl)

new_sp1 = lincomb(RS, A)
plot(new_sp1)

new_sp2 = lincomb(RS, A, reduced = FALSE)
plot(new_sp2)

#---------------------------------------------#
#--- Example with different support ranges ---#
#---------------------------------------------#
n=25; k=3
set.seed(5)
xi=sort(runif(n+2)); xi[1]=0; xi[n+2]=1
#Defining support ranges for three splines
supp=matrix(c(2,12,4,20,6,25),byrow=TRUE,ncol=2)
#Initial random matrices of the derivative for each spline
SS1=matrix(rnorm((supp[1,2]-supp[1,1]+1)*(k+1)),ncol=(k+1))
SS2=matrix(rnorm((supp[2,2]-supp[2,1]+1)*(k+1)),ncol=(k+1))
SS3=matrix(rnorm((supp[3,2]-supp[3,1]+1)*(k+1)),ncol=(k+1))
spl=construct(xi,k,SS1,supp[1,]) #constructing the first correct spline
nspl=construct(xi,k,SS2,supp[2,])
spl=gather(spl,nspl) #the second and the first ones
nspl=construct(xi,k,SS3,supp[3,])
spl=gather(spl,nspl) #the third is added

A = matrix(rnorm(3*2, mean = 2, sd = 100), ncol = 3)

new_sp1 = lincomb(spl, A) # based on reduced supports
plot(new_sp1)
new_sp2 = lincomb(spl, A, reduced = FALSE) # based on full support
plot(new_sp2) # new_sp1 and new_sp2 are same
```

```
#-----------------------------------------#
#--- Example with varying support sets ---#
#-----------------------------------------#
n=40; xi=seq(0,1,by=1/(n+1)); k=2;
support=list(matrix(c(2,9,15,24,30,37),ncol=2,byrow = TRUE))
sp=new("Splinets",knots=xi,smorder=k,supp=support)
m=sum(sp@supp[[1]][,2]-sp@supp[[1]][,1]+1) #the number of knots in the support
sp@der=list(matrix(rnorm(m*(k+1)),ncol=(k+1))) #the derivative matrix at random
sp1 = is.splinets(sp)[[2]] #the corrected vs. the original 'der' matrices

support=list(matrix(c(5,12,17,29),ncol=2,byrow = TRUE))
sp=new("Splinets",knots=xi,smorder=k,supp=support)
m=sum(sp@supp[[1]][,2]-sp@supp[[1]][,1]+1) #the number of knots in the support
sp@der=list(matrix(rnorm(m*(k+1)),ncol=(k+1))) #the derivative matrix at random
sp2 = is.splinets(sp)[[2]] #building a valid spline

spp = gather(sp1,sp2)

support=list(matrix(c(3,10,14,21,27,34),ncol=2,byrow = TRUE))
sp=new("Splinets",knots=xi,smorder=k,supp=support)
m=sum(sp@supp[[1]][,2]-sp@supp[[1]][,1]+1) #the number of knots in the support
sp@der=list(matrix(rnorm(m*(k+1)),ncol=(k+1))) #the derivative matrix at random
sp3 = is.splinets(sp)[[2]] #building a valid spline

spp = gather(spp, sp3)

plot(spp)
spp@supp #the supports

set.seed(5)
A = matrix(rnorm(3*4, mean = 2, sd = 100), ncol = 3)
new_sp1 = lincomb(spp, A) # based on reduced supports
plot(new_sp1)
new_sp1@supp #the support of the output from 'lincomb'

new_sp2 = lincomb(spp, A, reduced = FALSE) # based on full support
plot(new_sp2) # new_sp1 and new_sp2 are same
new_sp2@supp #the support of the output from 'lincomb' with full support computations

#-------------------------------------#
#--- Support needs some extra care ---#
#-------------------------------------#
set.seed(5)
n=53; k=4; xi=sort(runif(n+2)); xi[1]=0; xi[n+2]=1
supp1 = matrix(c(1, ceiling(n/2)+1), ncol = 2)
supp2 = matrix(c(ceiling(n/2)+1, n+2), ncol = 2)
S = matrix(rnorm(5*(ceiling(n/2)+1)), ncol = k+1)
a = construct(xi,k,S,supp = supp1) #constructing the first proper spline
S = matrix(rnorm(5*(ceiling(n/2)+1)), ncol = k+1)
b = construct(xi,k,S,supp = supp2) #constructing the first proper spline

sp = gather(a,b)
plot(sp)
```

```
# create a+b and a-b
s = lincomb(sp, matrix(c(1,1,1,-1), byrow = TRUE, nrow = 2))
plot(s)
s@supp

# Sum has smaller support than its terms
s1 = lincomb(s, matrix(c(1,1), nrow = 1), reduced = TRUE)
plot(s1)
s1@supp # lincomb based on support, the full support is reported
s2 = lincomb(s, matrix(c(1,1), nrow = 1), reduced = FALSE)
plot(s2)
s2@supp # lincomb using full der matrix

s3=lincomb(s, matrix(c(1,1), nrow = 1), reduced = FALSE, SuppExtr=FALSE)
s3@supp #the full range is reported as support

ES=exsupp(s1@der[[1]]) #correcting the matrix and the support
s1@der[[1]]=ES[[1]]
s1@supp[[1]]=ES[[2]]
plot(s1)
s1@supp[[1]]
```

`lines,Splinets-method` *Adding graphs of splines to a plot*

Description

A standard method of adding splines to an existing plot.

Usage

```
## S4 method for signature 'Splinets'
lines(x, sID = NULL, ...)
```

Arguments

`x`	`Splinets` object;
`sID`	vector, specifying indices of splines in the splinet object to be plotted;
`...`	other standard graphical parameters;

References

Liu, X., Nassar, H., Podgórski, K. "Dyadic diagonalization of positive definite band matrices and efficient B-spline orthogonalization." Journal of Computational and Applied Mathematics (2022) <https://doi.org/10.1016/j.cam.2022.114444>.

Podgórski, K. (2021) "`Splinets` – splines through the Taylor expansion, their support sets and orthogonal bases." <arXiv:2102.00733>.

Nassar, H., Podgórski, K. (2023) "Splinets 1.5.0 – Periodic Splinets." <arXiv:2302.07552>

See Also

plot,Splinets-method for graphical visualization of splines; evspline for evaluation of a Splinet-object;

Examples

```
#-----------------------------------------------------#
#------Adding spline lines to an existing graph-------#
#-----------------------------------------------------#
n=17; k=4; xi=sort(runif(n+2)); xi[1]=0; xi[n+2]=1
set.seed(5)
S=matrix(rnorm((n+2)*(k+1)),ncol=(k+1))
spl=construct(xi,k,S)
plot(spl,main="Mean Spline",lty=2,lwd=2)

RS=rspline(spl,5)
plot(RS,main="Random splines around the mean spline" )
lines(spl,col='red',lwd=4,lty=2)
```

---

plot,Splinets-method *Plotting splines*

Description

The method provides graphical visualization of a Splinets-class object. The method plot a Splinets in a cartesian or a polar coordinate if it is a regular splines or a periodic splines, respectively.

Usage

```
## S4 method for signature 'Splinets'
plot(
  object,
  x = NULL,
  sID = NULL,
  vknots = TRUE,
  type = "stnd",
  mrgn = 2,
  lwd = 2,
  ...
)
```

Arguments

object	Splinets object;
x	vector, specifying where the splines will be evaluated for the plots;
sID	vector, specifying indices of the splines to be plotted;

`vknots`	logic, indicates if auxiliary vertical lines will be added to highlight the positions of knots; The default is `TRUE`.
`type`	string, controls the layout of graphs; The following options are available • `"stnd"` – if `object@type="dspnt"` or `="spnt"`, then the plots are over the dyadic net of supports, other types of the bases are on a single plot with information about the basis printed out, • `"simple"` – all the objects are plotted in a single plot, • `"dyadic"` – if `object@type="sp"` is not true (unstructured collection of splines), then the plot is over the dyadic net of supports.
`mrgn`	number, specifying the margin size in the dyadic structure plot;
`lwd`	positive integer, the line width;
`...`	other standard graphical parameters can be passed;

Details

The standard method of plotting splines in a `Splinet`-object. It plots a single graph with all splines in the object except if the field `type` of the object represents a splinet. In the latter case, the default is the (dyadic) net plot of the basis. The string argument `type` can overide this to produce a plot that does not use the dyadic net. Most of the standard graphical parameters can be passed to this function.

Value

A plot visualizing a Splinet object. The entire set of splines will be displayed in a plot.

References

Liu, X., Nassar, H., Podgórski, K. "Dyadic diagonalization of positive definite band matrices and efficient B-spline orthogonalization." Journal of Computational and Applied Mathematics (2022) <https://doi.org/10.1016/j.cam.2022.114444>.

Podgórski, K. (2021) "`Splinets` – splines through the Taylor expansion, their support sets and orthogonal bases." <arXiv:2102.00733>.

Nassar, H., Podgórski, K. (2023) "Splinets 1.5.0 – Periodic Splinets." <arXiv:2302.07552>

See Also

`evspline` for manually evaluating splines in a `Splinet`-object; `Splinets-class` for the definition of the `Splinet`-class; `lines,Splinets-method` for adding graphs to existing plots;

Examples

```
#-----------------------------------------------------#
#-------------------Ploting splinets------------------#
#-----------------------------------------------------#
#Constructed splines

n=25; xi=sort(runif(n+2)); xi[1]=0; xi[n+2]=1; k=3
supp=list(t(c(2,12)),t(c(4,20)),t(c(6,25))) #defining support ranges for three splines
```

```
#Initial random matrices of the derivative for each spline
SS1=matrix(rnorm((supp[[1]][1,2]-supp[[1]][1,1]+1)*(k+1)),ncol=(k+1))
SS2=matrix(rnorm((supp[[2]][1,2]-supp[[2]][1,1]+1)*(k+1)),ncol=(k+1))
SS3=matrix(rnorm((supp[[3]][1,2]-supp[[3]][1,1]+1)*(k+1)),ncol=(k+1))

spl=construct(xi,k,SS1,supp[[1]]) #constructing the first correct spline
nspl=construct(xi,k,SS2,supp[[2]],'CRFC')
spl=gather(spl,nspl) #the second and the first ones
nspl=construct(xi,k,SS3,supp[[3]],'CRLC')
spl=gather(spl,nspl) #the third is added

plot(spl)
  plot(spl,sID=c(1,3))
plot(spl,sID=2)
t = seq(0,0.5,length.out = 1000)
plot(spl, t, sID = 1)

#Random splines
n=17; k=4; xi=sort(runif(n+2)); xi[1]=0; xi[n+2]=1
S=matrix(rnorm((n+2)*(k+1)),ncol=(k+1))
spl=construct(xi,k,S)
plot(spl,main="Mean Spline",lty=2,lwd=2,xlab='')

RS=rspline(spl,5)
plot(RS,main="Random splines around the mean spline",ylim=3*range(spl@der[[1]][,1]) )
lines(spl,col='red',lwd=4,lty=2)

#Periodic splines
xi = seq(0, 1, length.out = 25)
so = splinet(xi, periodic = TRUE)

plot(so$bs)
plot(so$os)
plot(so$bs,type= "dyadic")
plot(so$bs, sID=c(4,6))
plot(so$os, type="simple",sID=c(4,6))
```

---

project *Projecting into spline spaces*

Description

The projection of splines or functional data into the linear spline space spanned over a given set of knots.

Usage

```
project(
  fdsp,
  knots = NULL,
  smorder = 3,
  periodic = FALSE,
  basis = NULL,
  type = "spnt",
  graph = FALSE
)
```

Arguments

`fdsp`
: `Splinets`-object or a `n x (N+1)` matrix, a representation of `N` functions to be projected to the space spanned by a `Splinets`-basis over a specific set of knots; If the parameter is a `Splinets`-object containing `N` splines, then it is orthogonally projected or represented in the basis that is specified by other parameters. If the paramater is a matrix, then it is treated as `N` piecewise constant functions with the arguments in the first column and the corresponding values of the functions in the remaining `N` columns.

`knots`
: vector, the knots of the projection space, together with `smorder` fully characterizes the projection space; This parameter is overridden by the SLOT `basis@knots` of the `basis` input if this one is not `NULL`.

`smorder`
: integer, the order of smoothness of the projection space; This parameter is overridden by the SLOT `basis@smorder` of the `basis` input if this one is not `NULL`.

`periodic`
: logical, a flag to indicate if B-splines will be of periodic type or not; In the case of periodic splines, the arguments of the input and the knots need to be within [0,1] or, otherwise, an error occurs and a message advising the recentering and rescaling data is shown.

`basis`
: `Splinets`-object, the basis used for the representation of the projection of the input `fdsp`;

`type`
: string, the choice of the basis in the projection space used only if the `basis`-parameter is not given; The following choices are available

  - `'bs'` for the unorthogonalized B-splines,
  - `'spnt'` for the orthogonal splinet (the default),
  - `'gsob'` for the Gramm-Schmidt (one-sided) OB-splines,
  - `'twob'` for the two-sided OB-splines.

  The default is `'spnt'`.

`graph`
: logical, indicator if the illustrative plots are to be produced:

  - the splinet used in the projection(s) on the dyadic grid,
  - the coefficients of the projection(s) on the dyadic grid,
  - the input function(s),
  - the projection(s).

  ;

Details

The obtained coefficients $\mathbf{A} = (a_{ji})$ with respect to the basis allow to evaluate the splines S_j in the projection according to

$$S_j = \sum_{i=1}^{n-k-1} a_{ji} OB_i,\ j = 1, \dots, N,$$

where n is the number of the knots (including the endpoints), k is the spline smoothness order, N is the number of the projected functions and OB_i's consitute the considered basis. The coefficient for the splinet basis are always evaluated and thus, for example, `PFD=project(FD,knots);` `ProjDataSplines=lincomb(PFD$coeff,PFD$basis)` creates a `Splinets`-object made of the projections of the input functional data in `FD`. If the input parameter `basis` is given, then the function utilizes this basis and does not need to build it. However, if `basis` is the B-spline basis, then the B-spline orthogonalization is performed anyway, thus the computational gain is smaller than in the case when `basis` is an orthogonal basis.

Value

The value of the function is a list made of four elements

- `project$input` – `fdsp`, when the input is a `Splinets`-object or a matrix with the first column in an increasing order, otherwise it is the input numeric matrix after ordering according to the first column,
- `project$coeff` – `N x (n-k+1)` matrix of the coefficients of representation of the projection of the input in the splinet basis,
- `project$basis` – the spline basis,
- `projedt$sp` – the `Splinets`-object containing the projected splines.

References

Liu, X., Nassar, H., Podgórski, K. "Dyadic diagonalization of positive definite band matrices and efficient B-spline orthogonalization." Journal of Computational and Applied Mathematics (2022) <https://doi.org/10.1016/j.cam.2022.114444>.

Podgórski, K. (2021) "`Splinets` – splines through the Taylor expansion, their support sets and orthogonal bases." <arXiv:2102.00733>.

Nassar, H., Podgórski, K. (2023) "Splinets 1.5.0 – Periodic Splinets." <arXiv:2302.07552>

See Also

`refine` for embeding a `Splinets`-object into the space of splines with an extended set of knots; `lincomb` for evaluation of a linear combination of splines; `splinet` for obtaining the spline bases given the set of knots and the smootheness order;

Examples

```
#-------------------------------------------------#
#----Representing splines in the spline bases-----#
#-------------------------------------------------#
k=3 # order
```

```
n = 10 # number of the internal knots (excluding the endpoints)
xi = seq(0, 1, length.out = n+2)
set.seed(5)
S=matrix(rnorm((n+2)*(k+1)),ncol=(k+1))

spl=construct(xi,k,S)

plot(spl) # plotting a spline
spls=rspline(spl,5) # a random sample of splines

Repr=project(spl) #decomposition of splines into the splinet coefficients

Repr=project(spl, graph = TRUE) #decomposition of splines with the following graphs
                             #that illustrate the decomposition:
                             # 1) The orthogonal spine basis on the dyadic grid;
                            # 2) The coefficients of the projections on the dyadic grid;
                             # 3) The input splines;
                             # 4) The projections of the input.

Repr$coeff       #the coefficients of the decomposition
plot(Repr$sp) #plot of the reconstruction of the spline

plot(spls)
Reprs=project(spls,basis = Repr$basis) #decomposing splines using the available basis
plot(Reprs$sp)

Reprs2=project(spls,type = 'gsob') #using the Gram-Schmidt basis


#The case of the regular non-normalized B-splines:
Reprs3=project(spls,type = 'bs')
plot(Reprs3$basis)
gramian(Reprs3$basis,norm_only = TRUE) #the B-splines follow the classical definition and
                                       #thus are not normalized
plot(spls)

plot(Reprs3$basis) #Bsplines
plot(Reprs3$sp) #reconstruction using the B-splines and the decomposition

#a non-equidistant example
n=10; k=3
set.seed(5)
xi=sort(runif(n+2)); xi[1]=0; xi[n+2]=1
S=matrix(rnorm((n+2)*(k+1)),ncol=(k+1))
spl=construct(xi,k,S)
plot(spl)
spls=rspline(spl,5) # a random sample of splines
plot(spls)
Reprs=project(spls,type = 'twob') #decomposing using the two-sided orthogonalization
plot(Reprs$basis)
plot(Reprs$sp)

#The case of the regular non-normalized B-splines:
```

```
Reprs2=project(spls,basis=Reprs$basis)
plot(Reprs2$sp) #reconstruction using the B-splines and the decomposition

#-----------------------------------------------#
#-----Projecting splines into a spline space------#
#-----------------------------------------------#
k=3 # order
n = 10 # number of the internal knots (excluding the endpoints)
xi = seq(0, 1, length.out = n+2)
set.seed(5)
S=matrix(rnorm((n+2)*(k+1)),ncol=(k+1))

spl=construct(xi,k,S)

plot(spl) #the spline

knots=runif(8)
Prspl=project(spl,knots)

plot(Prspl$sp) #the projection spline
Rspl=refine(spl,newknots = knots) #embedding the spline to the common space
plot(Rspl)
RPspl=refine(Prspl$sp,newknots = xi)  #embedding the projection spline to the common space
plot(RPspl)
All=gather(RPspl,Rspl) #creating the Splinets-object with the spline and its projection
Rbasis=refine(Prspl$basis,newknots = xi) #embedding the basis to the common space
plot(Rbasis)

Res=lincomb(All,matrix(c(1,-1),ncol=2))
plot(Res)
gramian(Res,Sp2 = Rbasis) #the zero valued innerproducts -- the orthogonality of the residual spline

spls=rspline(spl,5) # a random sample of splines
Prspls=project(spls,knots,type='bs') #projection in the B-spline representation
plot(spls)
lines(Prspls$sp) #presenting projections on the common plot with the original splines

Prspls$sp@knots
Prspls$sp@supp

plot(Prspls$basis)   #Bspline basis


#An example with partial support

Bases=splinet(xi,k)

BS_Two=subsample(Bases$bs,c(2,length(Bases$bs@der)))
plot(BS_Two)
A=matrix(c(1,-2),ncol=2)
spl=lincomb(BS_Two,A)
plot(spl)
```

```
knots=runif(13)
Prspl=project(spl,knots)
plot(Prspl$sp)
Prspl$sp@knots
Prspl$sp@supp

#Using explicit bases

k=3 # order
n = 10 # number of the internal knots (excluding the endpoints)
xi = seq(0, 1, length.out = n+2)
set.seed(5)
S=matrix(rnorm((n+2)*(k+1)),ncol=(k+1))

spl=construct(xi,k,S)
spls=rspline(spl,5) # a random sample of splines

plot(spls)

knots=runif(20)
base=splinet(knots,smorder=k)
plot(base$os)

Prsps=project(spls,basis=base$os)
plot(Prsps$sp) #projection splines vs. the original splines
lines(spls)

#------------------------------------------------------#
#---Projecting discretized data into a spline space----#
#------------------------------------------------------#
k=3; n = 10; xi = seq(0, 1, length.out = n+2)
set.seed(5)
S=matrix(rnorm((n+2)*(k+1)),ncol=(k+1))
spl=construct(xi,k,S); spls=rspline(spl,10) # a random sample of splines

x=runif(50)
FData=evspline(spls,x=x) #discrete functional data

matplot(FData[,1],FData[,-1],pch='.',cex=3)
                   #adding small noise to the data
noise=matrix(rnorm(length(x)*10,0,sqrt(var(FData[,2]/10))),ncol=10)

FData[,-1]=FData[,-1]+noise

matplot(FData[,1],FData[,-1],pch='.',cex=3)

knots=runif(12)

DatProj=project(FData,knots)

lines(DatProj$sp) #the projections at the top of the original noised data

plot(DatProj$basis) #the splinet in the problem
```

```
#Adding knots to the projection space so that all data points are included
#in the range of the knots for the splinet basis

knots=c(-0.1,0.0,0.1,0.85, 0.9, 1.1,knots)

bases=splinet(knots)

DatProj1=project(FData,basis = bases$os)


matplot(FData[,1],FData[,-1],pch='.',cex=3)
lines(DatProj1$sp)

#Using the B-spline basis

knots=xi

bases=splinet(knots,type='bs')

DatProj3=project(FData,basis = bases$bs)

matplot(FData[,1],FData[,-1],pch='.',cex=3)
lines(DatProj3$sp)

DatProj4=project(FData,knots,k,type='bs') #this includes building the base of order 4

matplot(FData[,1],FData[,-1],pch='.',cex=3)
lines(DatProj4$sp)
lines(spls) #overlying the functions that the original data were built from

#Using two-sided orthonormal basis

DatProj5=project(FData,knots,type='twob')
matplot(FData[,1],FData[,-1],pch='.',cex=3)
lines(DatProj5$sp)
lines(spls)

#-------------------------------------------------#
#-----Projecting into a periodic spline space------#
#-------------------------------------------------#
#generating periodic splines
n=1# number of samples
k=3
N=3

n_knots=2^N*k-1 #the number of internal knots for the dyadic case
xi = seq(0, 1, length.out = n_knots+2)

so = splinet(xi,smorder = k, periodic = TRUE) #The splinet basis
stwo = splinet(xi,smorder = k,type='twob', periodic = TRUE) #The two-sided orthogonal basis

plot(so$bs,type='dyadic',main='B-Splines on dyadic structure') #B-splines on the dyadic graph
```

```
plot(stwo$os,main='Symmetric OB-Splines') #The two-sided orthogonal basis

plot(stwo$os,type='dyadic',main='Symmetric OB-Splines on dyadic structure')

# generating a periodic spline as a linear combination of the periodic splines
A1= as.matrix(c(1,0,0,0.7,0,0,0,0.8,0,0,0,0.4,0,0,0, 1, 0,0,0,0,0,1,0, .5),nrow= 1)
circular_spline=lincomb(so$os,t(A1))

plot(circular_spline)

#Graphical visualizations of the projections

Pro_spline=project(circular_spline,basis = so$os,graph = TRUE)
plot(Pro_spline$sp)

#-----------------------------------------------------------------#
#---Projecting discretized data into a periodic spline space----#
#-----------------------------------------------------------------#
nx=100 # number of discritization
n=1# number of samples
k=3
N=3

n_knots=2^N*k-1 #the number of internal knots for the dyadic case
xi = seq(0, 1, length.out = n_knots+2)

so = splinet(xi,smorder = k, periodic = TRUE)

hf=1/nx
grid=seq (hf , 1, by=hf) #grid
l=length(grid)
BB = evspline(so$os, x =grid)
fbases=matrix(c(BB[,2],BB[,5],BB[,9],BB[,13],BB[,17], BB[,23], BB[,25]), nrow = nx)

#constructing periodic data
f_circular=matrix(0,ncol=n+1,nrow=nx)
lambda=c(1,0.7,0.8,0.4, 1,1,.5)
f_circular[,1]= BB[,1]
f_circular[,2]= fbases%*%lambda

plot(f_circular[,1], f_circular[,2], type='l')

Pro=project(f_circular,basis = so$os)
plot(Pro$sp)
```

`refine` *Refining splines through adding knots*

Description

Any spline of a given order remains a spline of the same order if one considers it on a bigger set of knots than the original one. However, this embedding changes the `Splinets` representation of the so-refined spline. The function evaluates the corresponding `Splinets`-object.

Usage

```
refine(object, mult = 2, newknots = NULL)
```

Arguments

`object`	`Splinets`-object, the object to be represented as a `Splinets`-object over a refined set of knots;
`mult`	positive integer, refining rate; The number of the knots to be put equally spaced between the existing knots.
`newknots`	`m` vector, new knots; The knots do not need to be ordered and knots from the input `Splinets`-object knots are allowed since any ties are resolved.

Details

The function merges new knots with the ones from the input `object`. It utilizes `deriva()`-function to evaluate the derivative at the refined knots. It removes duplications of the refined knots, and account also for the not-fully supported case. In the case when the range of the additional knots extends beyond the knots of the input `Splinets`-object, the support sets of the output `Splinets`-object account for the smaller than the full support.

Value

A `Splinet` object with the new refined knots and the new matrix of derivatives is evaluated at the new knots combined with the original ones.

References

Liu, X., Nassar, H., Podgórski, K. "Dyadic diagonalization of positive definite band matrices and efficient B-spline orthogonalization." Journal of Computational and Applied Mathematics (2022) <https://doi.org/10.1016/j.cam.2022.114444>.

Podgórski, K. (2021) "`Splinets` – splines through the Taylor expansion, their support sets and orthogonal bases." <arXiv:2102.00733>.

Nassar, H., Podgórski, K. (2023) "Splinets 1.5.0 – Periodic Splinets." <arXiv:2302.07552>

See Also

`deriva` for computing derivatives at selected points; `project` for an orthogonal projection into a space of splines;

Examples

```
#-------------------------------------------------#
#----Refining splines - the full support case-----#
#-------------------------------------------------#
k=3 # order
n = 16 # number of the internal knots (excluding the endpoints)
xi = seq(0, 1, length.out = n+2)
set.seed(5)
S=matrix(rnorm((n+2)*(k+1)),ncol=(k+1))

spl=construct(xi,k,S)

plot(spl) # plotting a spline


rspl=refine(spl) # refining the equidistant by doubling its knots
plot(rspl)
rspl@equid  # the outcome is equidistant

#a non-equidistant case
n=17; k=4
xi=sort(runif(n+2)); xi[1]=0; xi[n+2]=1
S=matrix(rnorm((n+2)*(k+1)),ncol=(k+1))
spl=construct(xi,k,S)
plot(spl)
mult=3 #adding two knots between each subsequent pair of the original knots
rspl=refine(spl,mult)
is.splinets(rspl)
plot(rspl)

#adding specific knots
rspl=refine(spl,newknots=c(0.5,0.75))
rspl@knots
is.splinets(rspl)
plot(rspl)

#----------------------------------------------------#
#----Refining splines - the partial support case-----#
#----------------------------------------------------#

Bases=splinet(xi,k)
plot(Bases$bs)
Base=Bases$bs

BS_Two=subsample(Bases$bs,c(1,length(Base@der)))
plot(BS_Two)
A=matrix(c(1,-1),ncol=2)
spl=lincomb(BS_Two,A)

rspl=refine(spl) #doubling the number of knots
plot(rspl)
is.splinets(rspl)
```

```
rspl@supp #the support is evaluated
spl@supp

#The case of adding knots explicitely
BS_Middle=subsample(Bases$bs,c(floor(length(Base@der)/2)))
spls=gather(spl,BS_Middle)
plot(spls)

rspls=refine(spls, newknots=c(0.2,0.5,0.85)) #two splines with partial support sets
                                             #by adding three knots to B-splines
plot(rspls)

#----------------------------------------------------#
#------Refining splines over the larger range--------#
#----------------------------------------------------#

k=4 # order
n = 25 # number of the internal knots (excluding the endpoints)
xi = seq(0, 1, length.out = n+2)
S=matrix(rnorm((n+2)*(k+1)),ncol=(k+1))

spl=construct(xi,k,S)

plot(spl) # plotting a spline
newknots=c(-0.1,0.4,0.6,1.2)  #the added knots create larger range
rspl=refine(spl,newknots=newknots)
spl@supp #the original spline has the full support
rspl@supp #the embedded spline has partial support
spl@equid
rspl@equid
plot(rspl)
```

`rspline` *Random splines*

Description

The function simulates a random `Splinets`-object that is made of random splines with the center at the input spline and the matrix of derivatives has the added error term of the form

$$\mathbf{\Sigma}^{1/2}\mathbf{Z}\mathbf{\Theta}^{1/2},$$

where $\mathbf{Z}$ is a $(n+2) \times (k+1)$ matrix having iid standard normal variables as its entries, while $\mathbf{\Sigma}$ and $\mathbf{\Theta}$ are matrix parameters. This matrix error term is then corrected by one of the methods and thus resulting in a matrix of derivatives at knots corresponding to a valid spline.

Usage

```
rspline(S, N = 1, Sigma = NULL, Theta = NULL, mthd = "RRM")
```

Arguments

S	Splinets-object with n+2 knots and of the order of smoothness k, representing the center of randomly simulated splines; When the number of splines in the object is bigger than one, only the first spline in the object is used.
N	positive integer, size of the sample;
Sigma	matrix; • If (n+2)x(n+2) matrix, it controls correlations between derivatives of the same order at different knots. • If a positive number, it represents a diagonal (n+2)x(n+2) matrix with this number on the diagonal. • If a n+2 vector, it represents a diagonal (n+2)x(n+2) matrix with the vector entries on the diagonal. • If NULL (default) represents the identity matrix.
Theta	matrix; • If (k+1)x(k+1), this controls correlations between different derivatives at each knot. • If a positive number, it represents a diagonal matrix with this number on the diagonal. • If a k+1 vector, it represents a diagonal matrix with the vector entries on the diagonal. • If NULL (default), it represents the k+1 identity matrix;
mthd	string, one of the three methods: RCC, CR-LC, CR-FC, to adjust random error matrix so it corresponds to a valid spline;

Value

A Splinets-object that contains N generated splines constituting an iid sample of splines;

References

Liu, X., Nassar, H., Podgórski, K. "Dyadic diagonalization of positive definite band matrices and efficient B-spline orthogonalization." Journal of Computational and Applied Mathematics (2022) <https://doi.org/10.1016/j.cam.2022.114444>.

Podgórski, K. (2021) "Splinets – splines through the Taylor expansion, their support sets and orthogonal bases." <arXiv:2102.00733>.

Nassar, H., Podgórski, K. (2023) "Splinets 1.5.0 – Periodic Splinets." <arXiv:2302.07552>

See Also

is.splinets for diagnostics of the Splinets-objects; construct for constructing a Splinets-object; gather for combining Splinets-objects into a bigger object; subsample for subsampling Splinets-objects; plot,Splinets-method for plotting Splinets-objects;

Examples

```
#-----------------------------------------------------#
#-------Simulation of a standard random splinet-------#
#-----------------------------------------------------#
n=17; k=4; xi=sort(runif(n+2)); xi[1]=0; xi[n+2]=1
S=matrix(rnorm((n+2)*(k+1)),ncol=(k+1))
spl=construct(xi,k,S) #Construction of the mean spline

RS=rspline(spl)
graphsp=evspline(RS) #Evaluating the random spline
meansp=evspline(spl)
RS=rspline(spl,5) #Five more samples
graphsp5=evspline(RS)

m=min(graphsp[,2],meansp[,2],graphsp5[,2:6])
M=max(graphsp[,2],meansp[,2],graphsp5[,2:6])

plot(graphsp,type='l',ylim=c(m,M))
lines(meansp,col='red',lwd=3,lty=2) #the mean spline
for(i in 1:5){lines(graphsp5[,1],graphsp5[,i+1],col=i)}

#-----------------------------------------------------#
#------------Different construction method------------#
#-----------------------------------------------------#
RS=rspline(spl,8,mthd='CRLC'); graphsp8=evspline(RS)

m=min(graphsp[,2],meansp[,2],graphsp8[,2:6])
M=max(graphsp[,2],meansp[,2],graphsp8[,2:6])

plot(meansp,col='red',type='l',lwd=3,lty=2,ylim=c(m,M)) #the mean spline
for(i in 1:8){lines(graphsp8[,1],graphsp8[,i+1],col=i)}

#-----------------------------------------------------#
#-------Simulation of with different variances--------#
#-----------------------------------------------------#
Sigma=seq(0.1,1,n+2);Theta=seq(0.1,1,k+1)
RS=rspline(spl,N=10,Sigma=Sigma) #Ten samples
RS2=rspline(spl,N=10,Sigma=Sigma,Theta=Theta) #Ten samples
graphsp10=evspline(RS); graphsp102=evspline(RS2)

m=min(graphsp[,2],meansp[,2],graphsp10[,2:10])
M=max(graphsp[,2],meansp[,2],graphsp10[,2:10])

plot(meansp,type='l',ylim=c(m,M),col='red',lwd=3,lty=2)
for(i in 1:10){lines(graphsp10[,1],graphsp10[,i+1],col=i)}

m=min(graphsp[,2],meansp[,2],graphsp102[,2:10])
M=max(graphsp[,2],meansp[,2],graphsp102[,2:10])

plot(meansp,type='l',ylim=c(m,M),col='red',lwd=3,lty=2)
for(i in 1:10){lines(graphsp102[,1],graphsp102[,i+1],col=i)}
```

```
#-----------------------------------------------------#
#-------Simulation for the mean spline to be----------#
#------=----defined on incomplete supports------------#
#-----------------------------------------------------#
n=43; xi=seq(0,1,by=1/(n+1)); k=3; xi=sort(runif(n+2)); xi[1]=0; xi[n+2]=1;
support=list(matrix(c(2,14,25,43),ncol=2,byrow = TRUE))
ssp=new("Splinets",knots=xi,supp=support) #with partial support
nssp=is.splinets(ssp)$robject
nssp@smorder=3 #changing the order of the 'Splinets' object
m=sum(nssp@supp[[1]][,2]-nssp@supp[[1]][,1]+1) #the number of knots in the support
nssp@der=list(matrix(rnorm(m*(k+1)),ncol=(k+1)))  #the derivative matrix at random
spl=is.splinets(nssp)$robject
RS=rspline(spl,Sigma=0.05,Theta=c(1,0.5,0.3,0.05))
graphsp=evspline(RS);
meansp=evspline(spl)

m=min(graphsp[,2],meansp[,2],graphsp5[,2:6])
M=max(graphsp[,2],meansp[,2],graphsp5[,2:6])

plot(graphsp,type='l',ylim=c(m,M))
lines(meansp,col='red',lwd=3,lty=2) #the mean spline
```

---

seq2dyad *Organizing indices in a spline basis in the net form*

Description

This auxiliary function generates the map between the sequential order and the dyadic net structure of a spline basis. It works only with indices so it can be utilized to any basis in the space of splines with the zero-boundary conditions. The function is useful for creating the dyadic structure of the graphs and whenever a reference to the k-tuples and the levels of support is needed.

Usage

```
seq2dyad(n_sp, k)
```

Arguments

n_sp positive integer, the number of splines to be organized into the dyadic net; The dyadic net does not need to be fully dyadic, i.e. n_sp does not need to be equal to $k2^n - 1$, where n is the number of the internal knots. See the references for more details.

k the size of a tuple in the dyadic net; It naturally corresponds to the smoothness order of splines for which the net is build.

Value

The double indexed list of single row matrices of positive integers in the range `1:n_sp`. Each vector has typically the length `k` and some of them may correspond to incomplete tuplets and thus can be shorter. The first index in the list points to the level in the dyadic structure, the second one to the the number of the tuplet at the given level. The integers in the vector pointed by the list correspond to the sequential index of the element belonging to this tuplet.

References

Liu, X., Nassar, H., Podgórski, K. "Dyadic diagonalization of positive definite band matrices and efficient B-spline orthogonalization." Journal of Computational and Applied Mathematics (2022) <https://doi.org/10.1016/j.cam.2022.114444>.

Podgórski, K. (2021) "`Splinets` – splines through the Taylor expansion, their support sets and orthogonal bases." <arXiv:2102.00733>.

Nassar, H., Podgórski, K. (2023) "Splinets 1.5.0 – Periodic Splinets." <arXiv:2302.07552>

See Also

`plot,Splinets-method` for plotting splinets in the dydadic graphical representation; `lincomb` for evaluation of a linear combination of splines; `refine` for refinment of a spline to a larger number of knots;

Examples

```
#-----------------------------------------------------#
#--The support layers of the dyadic structure of bases--#
#-----------------------------------------------------#
k=4 # order
n = 36 # number of the internal knots (excluding the endpoints)
xi = seq(0, 1, length.out = n+2)
spnt=splinet(xi,k)

plot(spnt$os)                #standard plotting
plot(spnt$bs,type='dyadic')  #dyadic format of plots

net=seq2dyad(n-k+1,k)  #retrieving dyadic structure
ind1=c(net[[4]][[1]],net[[4]][[2]])

plot(subsample(spnt$os,ind1))

ind2=c(net[[4]][[3]],net[[4]][[4]]) #the lowest support in the dyadic net

lines(subsample(spnt$bs,ind2))
```

splinet *B-splines, periodic B-splines and their orthogonalization*

Description

The B-splines (periodic B-splines) are either given in the input or generated inside the routine. Then, given the B-splines and the argument `type`, the routine additionally generates a `Splinets`-object representing an orthonormal spline basis obtained from a certain orthonormalization of the B-splines. Orthonormal spline bases are obtained by one of the following methods: the Gram-Schmidt method, the two-sided method, and/or the splinet algorithm, which is the default method. All spline bases are kept in the format of `Splinets`-objects.

Usage

```
splinet(
  knots = NULL,
  smorder = 3,
  type = "spnt",
  Bsplines = NULL,
  periodic = FALSE,
  norm = F
)
```

Arguments

`knots` `n+2` vector, the knots (presented in the increasing order); It is not needed, when `Bsplines` argumment is not `NULL`, in which the case the knots from `Bsplines` are inherited.

`smorder` integer, the order of the splines, the default is `smorder=3`; Again it is inherited from the `Bsplines` argumment if the latter is not `NULL`.

`type` string, the type of the basis; The following choices are available

- `'bs'` for the unorthogonalized B-splines,
- `'spnt'` for the orthogonal splinet (the default),
- `'gsob'` for the Gramm-Schmidt (one-sided) O-splines,
- `'twob'` for the two-sided O-splines.

`Bsplines` `Splinet`-object, the basis of the B-splines (if not `NULL`); When this argument is not `NULL` the first two arguments are not needed since they will be inherited from `Bsplines`.

`periodic` logical, a flag to indicate if B-splines will be of periodic type or not;

`norm` logical, a flag to indicate if the output B-splines should be normalized;

Details

The B-spline basis, if not given in the input, is computed from the following recurrent (with respect to the smoothness order of the B-splines) formula

$$B_{l,k}^{\boldsymbol{\xi}}(x) = \frac{x - \xi_l}{\xi_{l+k} - \xi_l} B_{l,k-1}^{\boldsymbol{\xi}}(x) + \frac{\xi_{l+1+k} - x}{\xi_{l+1+k} - \xi_{l+1}} B_{l+1,k-1}^{\boldsymbol{\xi}}(x), l = 0, \ldots, n-k.$$

The dyadic algorithm that is implemented takes into account efficiencies due to the equally space knots (exhibited in the Toeplitz form of the Gram matrix) only if the problem is fully dyadic, i.e. if the number of the internal knots is `smorder*2^N-1`, for some integer `N`. To utilize this efficiency it may be advantageous, for a large number of equally spaced knots, to choose them so that their number follows the fully dyadic form. An additional advantage of the dyadic form is the complete symmetry at all levels of the support. The algorithm works with both zero boundary splines and periodic splines.

Value

Either a list `list("bs"=Bsplines)` made of a single `Splinet`-object `Bsplines` when `type=='bs'`, which represents the B-splines (the B-splines are normalized or not, depending on the `norm`-flag), or a list of two `Splinets`-objects: `list("bs"=Bsplines,"os"=Splinet)`, where `Bsplines` are either computed (in the input `Bspline= NULL`) or taken from the input `Bspline` (this output will be normalized or not depending on the `norm`-flag), `Splinet` is the B-spline orthognalization determined by the input argument `type`.

References

Liu, X., Nassar, H., Podgórski, K. "Dyadic diagonalization of positive definite band matrices and efficient B-spline orthogonalization." Journal of Computational and Applied Mathematics (2022) <https://doi.org/10.1016/j.cam.2022.114444>.

Podgórski, K. (2021) "`Splinets` – splines through the Taylor expansion, their support sets and orthogonal bases." <arXiv:2102.00733>.

Nassar, H., Podgórski, K. (2023) "Splinets 1.5.0 – Periodic Splinets." <arXiv:2302.07552>

See Also

`project` for projecting into the functional spaces spanned by the spline bases; `lincomb` for evaluation of a linear combination of splines; `seq2dyad` for building the dyadic structure for a splinet of a given smoothness order; `plot,Splinets-method` for visualisation of splinets;

Examples

```
#--------------------------------------#
#----Splinet, equally spaced knots-----#
#--------------------------------------#
k=2 # order
n_knots = 5 # number of knots
xi = seq(0, 1, length.out = n_knots)

so = splinet(xi, k)
```

```
plot(so$bs) #Plotting B-splines
plot(so$os) #Plotting Splinet

#Verifying the orthogonalization
gm = gramian(so$os) #evaluation of the inner products
diag(gm)
sum(gm - diag(diag(gm)))

#An example of the dyadic structure with equally spaced knots
k=3
N=3
n_knots=2^N*k-1 #the number of internal knots for the dyadic case

xi = seq(0, 1, length.out = n_knots+2)

so = splinet(xi)

plot(so$bs,type="simple",vknots=FALSE,lwd=3) #Plotting B-splines in a single simple plot
plot(so$os,type="simple",vknots=FALSE,lwd=3)

plot(so$os,lwd=3,mrgn=2) #Plotting the splinet on the dyadic net of support intervals

so=splinet(xi, Bsplines=so$bs, type='gsob') #Obtaining the Gram-Schmidt orthogonalization
plot(so$os,type="simple",vknots=FALSE)      #Without computing B-splines again

so=splinet(xi, Bsplines=so$bs, type='twob') #Obtaining the symmetrize orthogonalization
plot(so$os,type="simple",vknots=FALSE)

#-------------------------------------#
#---Splinet, unequally spaced knots---#
#-------------------------------------#
n_knots=25

xi = c(0, sort(runif(n_knots)), 1)

sone = splinet(xi, k)

plot(sone$bs, type='dyadic') #Plotting B-splines
plot(sone$os) #Plotting Splinet

#Verifying the orthogonalization
gm = gramian(sone$os) #evaluation of the inner products
diag(gm)
sum(gm - diag(diag(gm)))

#-------------------------------------------#
#---Dyadic splinet, equally spaced knots---#
#-------------------------------------------#
k = 2 # order
N = 3 # support level
n_so = k*(2^N-1) # number of splines in a dyadic structure with N and k
n_knots = n_so + k + 1 # number of knots
xi = seq(0, 1, length.out = n_knots)
```

```
sodyeq = splinet(xi, k)

plot(sodyeq$bs) #Plotting B-splines
plot(sodyeq$os) #Plotting Splinet

#Verifying the orthogonalization
gm = gramian(sodyeq$os) #evaluation of the inner products
diag(gm)
sum(gm - diag(diag(gm)))

#--------------------------------------------#
#---Dyadic splinet, unequally spaced knots---#
#--------------------------------------------#
xi = c(0, sort(runif(n_knots)), 1)

sody = splinet(xi, k)


plot(sody$bs) #Plotting B-splines
plot(sody$os) #Plotting Splinet

#Verifying the orthogonalization
gm = gramian(sody$os) #evaluation of the inner products
diag(gm)
sum(gm - diag(diag(gm)))

#-----------------------------------------#
#---Bspline basis, equally spaced knots---#
#-----------------------------------------#
n = 15
xi = seq(0,1,length.out = n+2)
order = 2

bs = splinet(xi, order, type = 'bs')

plot(bs$bs)

#---------------------------------------------#
#---Bspline basis, non-equally spaced knots---#
#---------------------------------------------#
n = 6
xi = c(0,sort(runif(n)),1)
order = 3

so = splinet(xi, order, type = 'bs') #unnormalized version
plot(so$bs)

so1 = splinet(type='bs',Bsplines=so$bs,norm=TRUE) #normalized version
plot(so1$bs)

#--------------------------------------------------#
#---Gram-Schmidt osplines, equally spaced knots---#
```

```
#--------------------------------------------------#
so = splinet(xi, order,  type = 'gsob')

plot(so$bs)
plot(so$os)

#Using the previously generated B-splines and normalizing them
so1 = splinet(Bsplines=so$bs, type = "gsob",norm=TRUE)

plot(so1$bs) #normalized B-splines
plot(so1$os) #the one sided osplines

gm = gramian(so1$os) #evaluation of the inner products
diag(gm)
sum(gm - diag(diag(gm))) #verification of the orghonoalization of the matrix

#------------------------------------------------------#
#---Gram-Schmidt osplines, non-equally spaced knots---#
#------------------------------------------------------#

so = splinet(Bsplines=sody$bs, type = 'gsob') #previously genereted Bsplines

plot(so$bs)
plot(so$os)

gm = gramian(so$os)
diag(gm)
sum(gm - diag(diag(gm)))

#----------------------------------------------#
#---Twosided osplines, equally spaced knots---#
#----------------------------------------------#

so = splinet(Bsplines=bs$bs, type = 'twob')
plot(so$os)

gm = gramian(so$os) #verification of the orthogonality
diag(gm)
sum(gm - diag(diag(gm)))

#--------------------------------------------------#
#---Twosided osplines, non equally spaced knots---#
#--------------------------------------------------#

so = splinet(Bsplines=sody$bs, type = 'twob')
plot(so$os)

gm = gramian(so$os) #verification of the orthogonality
diag(gm)
sum(gm - diag(diag(gm)))

#---------------------------------------------#
#---Periodic splinet, equally spaced knots---#
```

```
#--------------------------------------------#
k=2 # order
n_knots = 12 # number of knots
xi = seq(0, 1, length.out = n_knots)

so = splinet(xi, k, periodic = TRUE)

plot(so$bs) #Plotting B-splines
plot(so$os) #Plotting Splinet

#Verifying the orthogonalization
gm = gramian(so$os) #evaluation of the inner products
diag(gm)
sum(gm - diag(diag(gm)))

#An example of the dyadic structure with equally spaced knots
k=3
N=3
n_knots=2^N*k-1 #the number of internal knots for the dyadic case

xi = seq(0, 1, length.out = n_knots+2)

so = splinet(xi, periodic = TRUE)

plot(so$bs,type="simple") #Plotting B-splines in a single simple plot
plot(so$os,type="simple")

plot(so$os) #Plotting the splinet on the dyadic net of support intervals

so=splinet(xi, Bsplines=so$bs, type='gsob') #Obtaining the Gram-Schmidt orthogonalization
plot(so$os,type="simple")      #Without computing B-splines again

so=splinet(xi, Bsplines=so$bs , type='twob') #Obtaining the symmetrize orthogonalization
plot(so$os,type="simple")


#-----------------------------------#
#---Splinet, unequally spaced knots---#
#-----------------------------------#
n_knots=25

xi = c(0, sort(runif(n_knots)), 1)

sone = splinet(xi, k, periodic = TRUE)

plot(sone$bs, type='dyadic') #Plotting B-splines
plot(sone$os) #Plotting Splinet

#Verifying the orthogonalization
gm = gramian(sone$os) #evaluation of the inner products
diag(gm)
sum(gm - diag(diag(gm)))
```

```
#------------------------------------------#
#---Dyadic splinet, equally spaced knots---#
#------------------------------------------#
k = 2 # order
N = 3 # support level
n_so = k*(2^N-1) # number of splines in a dyadic structure with N and k
n_knots = n_so + k + 1 # number of knots
xi = seq(0, 1, length.out = n_knots)

sodyeq = splinet(xi, k, periodic = TRUE)

plot(sodyeq$bs) #Plotting B-splines
plot(sodyeq$os) #Plotting Splinet

#Verifying the orthogonalization
gm = gramian(sodyeq$os) #evaluation of the inner products
diag(gm)
sum(gm - diag(diag(gm)))

#--------------------------------------------#
#---Dyadic splinet, unequally spaced knots---#
#--------------------------------------------#
xi = c(0, sort(runif(n_knots)), 1)

sody = splinet(xi, k, periodic = TRUE)


plot(sody$bs) #Plotting B-splines
plot(sody$os) #Plotting Splinet

#Verifying the orthogonalization
gm = gramian(sody$os) #evaluation of the inner products
diag(gm)
sum(gm - diag(diag(gm)))
```

`Splinets-class` *The class to represent a collection of splines*

Description

The main class in the `splinets`-package used for representing a collection of splines.

Value

running `new("Splinets")` return an object that belongs to the class `Splinets`, with the initialization of the default values for the fields.

Slots

`knots` numeric `n+2` vector, a vector of `n+2` knot locations presented in the increasing order and without ties;

`smorder` non-negative integer, the smoothnes order of the splines, i.e. the highest order of non-zero derivative;

`equid` logical, indicates if the knots are equidistant; Some computations in the equidistant case are simpler so this information helps to account for it.

`supp` list (of matrices),

- `length(supp)==0` – the full support set for all splines,
- `length(supp)==N` – support sets for `N` splines;

If non-empty, a list containing `Nsupp x 2` matrices (of positive integers). If `Nsupp` is equal to one it should be a row matrix (not a vector). The rows in the matrices, `supp[[i]][l,]`, `l in 1:Nsupp` represents the indices of the knots that are the endpoints of the intervals in the support sets. Each of the support set is represented as a union of disjoint `Nsupp` intervals, with knots as the endpoints. Outside the set (support), the spline vanishes. Each matrix in this list is ordered so the rows closer to the top correspond to the intervals closer to the LHS end of the support.

`der` list (of matrices); a list of the length `N` containing `sum(supp[[i]][,2]-supp[[i]][,1]+1) x (smorder+1)` matrices, where `i` is the index running through the list. Each matrix in the list includes the values of the derivatives at the knots in the support of the corresponding spline.

`taylor` `(n+1) x (smorder+1)`, if `equid=FALSE`, or `1 x (smorder+1)` if `equid=TRUE`, columnwise vectors of the Taylor expansion coefficients at the knots; Vectors instead of matrices are recognized properly. The knot and order dependent matrix of rows of coefficients used in the Taylor expansion of splines. Once evaluated it can be used in computations for any spline of the given order over the given knots. The columns of this matrix are used for evaluation of the values of the splines in-between knots, see the references for further details.

`type` string, one of the following character strings: `bs`,`gsob`,`twob`,`dspnt`,`spnt`,`sp`; The default is `sp` which indicates any unstructured collection of splines. The rest of the strings indicate different *spline bases*:

- `bs` for B-splines,
- `gsob` for Gram-Schmidt O-splines,
- `twob` for two-sided O-splines,
- `dspnt` for a fully dyadic splinet,
- `spnt` for a non-dyadic splinet.

`periodic` logical, indicates if the B-splines are periodic or not.

`epsilon` numeric (positive), an accuracy used to detect a problem with the conditions required for the matrix of the derivatives (controls relative deviation from the conditions);

References

Liu, X., Nassar, H., Podgórski, K. "Dyadic diagonalization of positive definite band matrices and efficient B-spline orthogonalization." Journal of Computational and Applied Mathematics (2022) <https://doi.org/10.1016/j.cam.2022.114444>.

Podgórski, K. (2021) "`Splinets` – splines through the Taylor expansion, their support sets and orthogonal bases." <arXiv:2102.00733>.

Nassar, H., Podgórski, K. (2023) "Splinets 1.5.0 – Periodic Splinets." <arXiv:2302.07552>

See Also

`is.splinets` for evaluation of a `Splinets`-object; `construct` for constructing a `Splinets`-object; `plot,Splinets-method` for plotting methods for `Splinets`-objects;

Examples

```
#-------------------------------------------------------------#
#-------Generating an object from the class 'Splinets'--------#
#-------------------------------------------------------------#
#The most generic generation of an object of class 'Splinets':
sp=new("Splinets") #a generic format for 'Splinets' object
sp
#The most important SLOTs of 'Splinets' - the default values
sp@knots
sp@smorder
sp@der
sp@supp

set.seed(5); n=13; xi=sort(runif(n+2)); xi[1]=0;xi[n+2]=1
sp@knots=xi #randomly assigned knots

#Changing the order of
#smoothness and intializing Taylor coefficients
ssp=new("Splinets",knots=xi,smorder=2)
ssp@taylor

#Equidistant case
ssp=new("Splinets",knots=seq(0,1,1/(n+1)),smorder=3)
ssp@taylor
ssp@equid
```

`subsample` *Subsampling from a set of splines*

Description

The function constructs a `Splinets`-object that is made of subsampled elements of the input `Splinets`-object. The input objects have to be of the same order and over the same knots.

Usage

```
subsample(Sp, ss)
```

Arguments

Sp	Splinets-object, a collection of s splines;

ss	vector of integers, the coordinates from 1:s;

Details

The output Splinet-object made of subsampled splines is always is of the regular type, i.e. SLOT type='sp'.

Value

An Splinets-object containing length(ss) splines that are selected from the input object./

References

Liu, X., Nassar, H., Podgórski, K. "Dyadic diagonalization of positive definite band matrices and efficient B-spline orthogonalization." Journal of Computational and Applied Mathematics (2022) <https://doi.org/10.1016/j.cam.2022.114444>.

Podgórski, K. (2021) "Splinets – splines through the Taylor expansion, their support sets and orthogonal bases." <arXiv:2102.00733>.

Nassar, H., Podgórski, K. (2023) "Splinets 1.5.0 – Periodic Splinets." <arXiv:2302.07552>

See Also

is.splinets for diagnostic of Splinets-objects; construct for constructing such a Splinets-object; gather for combining Splinets-objects; refine for refinment of a spline to a larger number of knots; plot,Splinets-method for plotting Splinets-objects;

Examples

```
#-----------------------------------------------------#
#--------------------Subsampling--------------------#
#-----------------------------------------------------#

#Example with different support ranges, the 3rd order
n=25; xi=sort(runif(n+2)); xi[1]=0; xi[n+2]=1; k=3
supp=list(t(c(2,12)),t(c(4,20)),t(c(6,25))) #defining support ranges for three splines

#Initial random matrices of the derivative for each spline
set.seed(5)
SS1=matrix(rnorm((supp[[1]][1,2]-supp[[1]][1,1]+1)*(k+1)),ncol=(k+1))
SS2=matrix(rnorm((supp[[2]][1,2]-supp[[2]][1,1]+1)*(k+1)),ncol=(k+1))
SS3=matrix(rnorm((supp[[3]][1,2]-supp[[3]][1,1]+1)*(k+1)),ncol=(k+1))

spl=construct(xi,k,SS1,supp[[1]]) #constructing the first correct spline
nspl=construct(xi,k,SS2,supp[[2]],'CRFC')

#See 'gather' function for more details on what follows
spl=gather(spl,nspl) #the second and the first ones
nspl=construct(xi,k,SS3,supp[[3]],'CRLC')
```

```
spl=gather(spl,nspl) #the third is added

#Replicating by subsampling with replacement
sz=length(spl@der)
ss=sample(1:sz,size=10,rep=TRUE)

spl=subsample(spl,ss)
is.splinets(spl)[[1]]

spl@supp
spl@der

#Subsampling without replacements
ss=c(3,8,1)
sspl=subsample(spl,ss)

sspl@supp
sspl@der

is.splinets(sspl)[[1]]

#A single spline sampled from a 'Splinets' object
is.splinets(subsample(sspl,1))
```

---

sym2one *Switching between representations of the matrices of derivatives*

Description

A technical but useful transformation of the matrix of derivatives form the one-sided to symmetric representations, or a reverse one. It allows for switching between the standard representation of the matrix of the derivatives for `Splinets` which is symmetric around the central knot(s) to the one-sided that yields the RHS limits at the knots, which is more convenient for computations.

Usage

```
sym2one(S, supp = NULL, inv = FALSE)
```

Arguments

`S`	`(m+2) x (k+1)` numeric matrix, the derivatives in one of the two representations;
`supp`	`(Nsupp x 2)` or `NULL` matrix, row-wise the endpoint indices of the support intervals; If it is equal to `NULL` (which is also the default), then the full support is assumed.
`inv`	logical; If `FALSE` (default), then the function assumes that the input is in the symmetric format and transforms it to the left-to-right format. If `TRUE`, then the inverse transformation is applied.

Details

The transformation essentially changes only the last column in S, i.e. the highest (discontinuous) derivatives so that the one-sided representation yields the right-hand-side limit. It is expected that the number of rows in S is the same as the total size of the support as indicated by supp, i.e. if supp!=NULL, then sum(supp[,2]-supp[,1]+1)=m+2. If the latter is true, than all derivative sub-matrices of the components in S will be reversed. However, this condition formally is not checked in the code, which may lead to switch of the representations only for parts of the matrix S.

Value

A matrix that is the respective transformation of the input.

References

Liu, X., Nassar, H., Podgórski, K. "Dyadic diagonalization of positive definite band matrices and efficient B-spline orthogonalization." Journal of Computational and Applied Mathematics (2022) <https://doi.org/10.1016/j.cam.2022.114444>.

Podgórski, K. (2021) "Splinets – splines through the Taylor expansion, their support sets and orthogonal bases." <arXiv:2102.00733>.

Nassar, H., Podgórski, K. (2023) "Splinets 1.5.0 – Periodic Splinets." <arXiv:2302.07552>

See Also

Splinets-class for the description of the Splinets-class; is.splinets for diagnostic of Splinets-objects;

Examples

```
#-----------------------------------------------------#
#-------Representations of derivatives at knots-------#
#-----------------------------------------------------#
n=10; k=3; xi=seq(0,1,by=1/(n+1)) #the even number of equally spaced knots
set.seed(5)
S=matrix(rnorm((n+2)*(k+1)),ncol=(k+1))
spl=construct(xi,k,S) #construction of a spline
a=spl@der[[1]]
b=sym2one(a)
aa=sym2one(b,inv=TRUE) # matrix 'aa' is the same as 'a'

n=11; xi2=seq(0,1,by=1/(n+1)) #the odd number of knots case
S2=matrix(rnorm((n+2)*(k+1)),ncol=(k+1))
spl2=construct(xi2,k,S2) #construction of a spline
a2=spl2@der[[1]]
b2=sym2one(a2)
aa2=sym2one(b2, inv=TRUE) # matrix 'aa2' is the same as 'a2'

#-----------------------------------------------------#
#--------------More complex support sets--------------#
#-----------------------------------------------------#
#Zero order splines, non-equidistant case, support with three components
```

```
n=43; xi=seq(0,1,by=1/(n+1)); k=3; xi=sort(runif(n+2)); xi[1]=0; xi[n+2]=1;
support=list(matrix(c(2,14,17,30,32,43),ncol=2,byrow = TRUE))
#Third order splines
ssp=new("Splinets",knots=xi,supp=support,smorder=k) #with partial support

m=sum(ssp@supp[[1]][,2]-ssp@supp[[1]][,1]+1) #the total number of knots in the support
ssp@der=list(matrix(rnorm(m*(k+1)),ncol=(k+1)))  #the derivative matrix at random
IS=is.splinets(ssp)
IS$robject@der
IS$robject@supp
b=sym2one(IS$robject@der[[1]],IS$robject@supp[[1]]) #the RHS limits at the knots
a=sym2one(b,IS$robject@supp[[1]],inv=TRUE) #is the same as the SLOT supp in IS@robject
```

---

tire *Data on tire responses to a rough road profile*

Description

These are simulated data of tire responses to a rough road at the high-transient event. The simulations have been made based on the fit of the so-called Slepian model to a non-Gaussian rough road profile. Further details can be found in the reference. The responses provided are measured at the wheel and thus describing the tire response. There are 100 functional measurments, kept column-wise in the matrix. Additionally, the time instants of the measurements are given as the first column in the matrix. Since the package uses the so-called "lazy load", the matrix is directly available without an explicit load of the data. This means that `data(tire)` does not need to be invoked. The data were saved using `compress='xz'` option, which requires 3.5 or higher version of R. The data are uploaded as a dataframe, thus `as.matrix(tire)` is needed if the matrix form is required.

Usage

```
data(tire)
```

Format

numerical `4095 x 101` dataframe: `tire`

References

Podgórski, K, Rychlik, I. and Wallin, J. (2015) Slepian noise approach for gaussian and Laplace moving average processes. Extremes, 18(4):665–695, <doi:10.1007/s10687-015-0227-z>.

See Also

truck for a related dataset;

Examples

```
#----------------------------------------------------#
#----------- Plotting the trucktire data -------------#
#----------------------------------------------------#

#Activating data:
 data(tire)
 data(truck)

 matplot(tire[,1],tire[,2:11],type='l',lty=1) #ploting the first 10 tire responses

 matplot(truck[,1],truck[,2:11],type='l',lty=1) #ploting the first 10 truck responses

 #Projecting truck data into splinet bases
 knots1=seq(0,50, by=2)
 Subtruck= truck[2048:3080,] # selecting the truck data that in the interval[0,50]
 TruckProj=project(as.matrix(Subtruck),knots1)

 MeanTruck=matrix(colMeans(TruckProj$coeff),ncol=dim(TruckProj$coeff)[2])
 MeanTruckSp=lincomb(TruckProj$basis,MeanTruck)

 plot(MeanTruckSp) #the mean spline of the projections

 plot(TruckProj$sp,sID=1:10) #the first ten projections of the functional data

 Sigma=cov(TruckProj$coeff)
 Spect=eigen(Sigma,symmetric = TRUE)

 plot(Spect$values, type ='l',col='blue', lwd=4 ) #the eigenvalues

 EigenTruckSp=lincomb(TruckProj$basis,t(Spect$vec))
 plot(EigenTruckSp,sID=1:5) #the first five largest eigenfunctions
```

---

truck *Data on truck responses to a rough road profile*

Description

These are simulated data of truck responses to a rough road at the high transient event. The simulations have been made based on the fit of the so-called Slepian model to a non-Gaussian rough road profile. Details can be found in the reference. The responses provided are at the driver seat. There are 100 functional measurments, kept column-wise in the matrix. Additionally, the time instants of the measurements are given as the first column in the matrix. Since the package uses the so-called "lazy load", the matrix is directly available without an explicit load of the data, thus `data(truck)` does not need to be invoked. Data were saved using `compress='xz'` option, which requires 3.5 or higher version of R. The data are uploaded as a dataframe, thus `as.matrix(tire)` is needed if the matrix form is required.

Usage

```
data(truck)
```

Format

numerical 4095 x 101 dataframe: truck

References

Podgórski, K, Rychlik, I. and Wallin, J. (2015) Slepian noise approach for gaussian and Laplace moving average processes. Extremes, 18(4):665–695, <doi:10.1007/s10687-015-0227-z>.

See Also

tire for a related dataset;

Examples

```
#-----------------------------------------------------#
#----------- Plotting the trucktire data -------------#
#-----------------------------------------------------#

#Activating data:
 data(tire)
 data(truck)

 matplot(tire[,1],tire[,2:11],type='l',lty=1) #ploting the first 10 tire responses

 matplot(truck[,1],truck[,2:11],type='l',lty=1) #ploting the first 10 truck responses

 #Projecting truck data into splinet bases
 knots1=seq(0,50, by=2)
 Subtruck= truck[2048:3080,] # selecting the truck data that in the interval[0,50]
 TruckProj=project(as.matrix(Subtruck),knots1)

 MeanTruck=matrix(colMeans(TruckProj$coeff),ncol=dim(TruckProj$coeff)[2])
 MeanTruckSp=lincomb(TruckProj$basis,MeanTruck)

 plot(MeanTruckSp) #the mean spline of the projections

 plot(TruckProj$sp,sID=1:10) #the first ten projections of the functional data

 Sigma=cov(TruckProj$coeff)
 Spect=eigen(Sigma,symmetric = TRUE)

 plot(Spect$values, type ='l',col='blue', lwd=4 ) #the eigenvalues

 EigenTruckSp=lincomb(TruckProj$basis,t(Spect$vec))
 plot(EigenTruckSp,sID=1:5) #the first five largest eigenfunctions
```

wind *Data on wind direction and speed.*

Description

NASA/POWER CERES/MERRA2 Native Resolution Hourly Data

- Dates: 01/01/2015 through 03/05/2015
- Location: Latitude 25.7926 Longitude -80.3239
- Elevation from MERRA-2: Average for 0.5 x 0.625 degree lat/lon region = 5.4 meters

Data frame fields:

- `YEAR` – Year of a measurement
- `MO` – Month of a measurement
- `DY` – Day of a measurement
- `HR` – Hour of a measurement
- `WD10M` – MERRA-2 Wind Direction at 10 Meters (Degrees)
- `WS50M` – MERRA-2 Wind Speed at 50 Meters (m/s)
- `WD50M` – MERRA-2 Wind Direction at 50 Meters (Degrees)
- `WS10M` – MERRA-2 Wind Speed at 10 Meters (m/s)

Usage

```
data(wind)
```

Format

numerical `1536 x 8` dataframe: `wind`

References

The data was obtained from the National Aeronautics and Space Administration (NASA) Langley Research Center (LaRC) Prediction of Worldwide Energy Resource (POWER) Project funded through the NASA Earth Science/Applied Science Program. https://power.larc.nasa.gov/data-access-viewer/

Examples

```
#-----------------------------------------------#
#----------- Plotting the Wind data -------------#
#-----------------------------------------------#

data(wind) #activating the data

wind1=wind[,-1] #Removing YEAR as irrelevant
```

```
#Transforming data to daily with the periodic form, i.e. the arguments in [0,1],
#which is required in the periodic case.

numbdays=length(wind1[,1])/24

Days=vector(mode='list', length=numbdays)

for(i in 1:numbdays){
  Days[[i]]=wind1[i*(1:24),]
  Days[[i]][,c(4,6)]=Days[[i]][,c(4,6)]/360 #the direction in [0,1]
}

#Raw discretized data for the first day

par(mfrow=c(2,2))
hist(Days[[1]][,4],xlim=c(0,1),xlab='Wind direction',main='First day 10[m]')
hist(Days[[1]][,6],xlim=c(0,1),xlab='Wind direction',main='First day 50[m]')

plot(Days[[1]][,4],Days[[1]][,5],xlim=c(0,1),pch='.',cex=4,xlab='Wind direction',ylab='Wind speed')
plot(Days[[1]][,6],Days[[1]][,7],xlim=c(0,1),pch='.',cex=4,xlab='Wind direction',ylab='Wind speed')

#First Day Data:
#Projections of the histograms to the periodic spline form

FirstDayDataF1=cbind(hist(Days[[1]][,4],xlim=c(0,1),breaks=seq(0,1,by=0.1))$mids,
hist(Days[[1]][,4],xlim=c(0,1),breaks=seq(0,1,by=0.1))$counts)


k=3
N=2
n_knots=2^N*k-1 #the number of internal knots for the dyadic case
xi = seq(0, 1, length.out = n_knots+2)
#Note that the range of the argument is assumed to be between 0 and 1

PrF1=project(FirstDayDataF1,xi,periodic = TRUE, graph = TRUE)

F1=PrF1$sp #The first day projection of the direction histogram at 10[m]

#Projections of the scatterplots to the periodic spline form
#The bivariate sampl
FirstDayDataF1V1=as.matrix(Days[[1]][,4:5]) #we note that wind directions are scaled but not ordered

#Padding the data with zeros as the sampling frequency is not sufficiently dense over [0,1]
FirstDayDataF1V1=rbind(FirstDayDataF1V1,cbind(seq(0,1,by=1/24),rep(0,25)))

#Another knot selection with more knots but still dyadic case
k=4
N=3
n_knots2=2^N*k-1 #the number of internal knots for the dyadic case
xi2 = seq(0, 1, length.out = n_knots2+2)

#For illustration one can plot the B-splines and the corresponding splinet
```

```
so = splinet(xi2,smorder = k, periodic = TRUE,norm = TRUE)

plot(so$bs)
plot(so$bs,type='dyadic') #To facilitate the comparison with the splinet better
                          #one can choose the dyadic grapph
plot(so$os)


#Projecting direction/wind data onto splines
PrS1=project(FirstDayDataF1V1,xi2,smorder=k,periodic = TRUE, graph = TRUE)

S1=PrS1$sp

#the next 7 days
days= 7
#Transforming to the periodic data

#The direction histogram
for(i in 2:days){
  DataF1=cbind(hist(Days[[i]][,4],plot=FALSE,breaks=seq(0,1,by=0.1))$mids,
               hist(Days[[i]][,4],plot=FALSE,breaks=seq(0,1,by=0.1))$counts)
  PrF1=project(DataF1,xi,periodic = TRUE)

 F1=gather(F1,PrF1$sp) #Collecting projections of daily wind-direction histograms at 10[m]

}

plot(F1) #plot of all daily functional data wind direction distributions


#Wind direction vs speed data at 10[m]
for(i in 2:days){
 DataF1V1=as.matrix(Days[[i]][,4:5]) #we note that wind directions are scaled but not ordered

 #Padding the data with zeros as the sampling frequency is not sufficiently dense over [0,1]
  DataF1V1=rbind(DataF1V1,cbind(seq(0,1,by=1/24),rep(0,25)))

  PrS1=project(DataF1V1,xi2,smorder=k,periodic = TRUE)

 S1=gather(S1,PrS1$sp) #Collecting projections of daily wind-direction histograms at 10[m]

}

plot(S1) #plot of all daily functional data wind speed at wind direction

#Computing means of the data
A=matrix(rep(1/days,days),ncol=days)

MeanF1=lincomb(F1,A)
plot(MeanF1)

MeanS1=lincomb(S1,A)
plot(MeanS1)
```

Index