Highly nonlinear approximations for signal representation
Manual for MATLAB routines

James BOWLEY, Laura REBOLLO-NEIRA and Zhiqiang XU
Aston University, Birmingham, B4 7ET, UK

`http://www.nonlinear-approx.info`

This project is supported by EPSRC (EP/D062632/1).

Engineering and Physical Sciences
Research Council

Contents

Preface

This document was written as an users guide to the computational tools delivered by the EPSRC funded project "Highly nonlinear approximations for sparse signal representation". More information about the project and tutorial material related to the routines of this manual are given in the website http://www.nonlinear-approx.info.

The project followed on of the previous EPSRC funded project "Biorthogonal techniques for optimal signal represeation. Thanks are due to Miroslav Andrle for leaving the material of that project well organized, which facilitated the continuation of the work.

We would like to thank to Andreas Hartmann for his `M2TEX` script which has been used for generating most of this manual from our MATLAB sources.

Part I

Pursuits

Chapter 1

Pursuits

1.1 Function-Summary

BOOMP	Backward-Optimized Orthogonal Matching Pursuit
BOOMPQ	Backward-Optimized Orthogonal Matching and gives the orthonormal basis
KSwapping	extends Swapping to considering K swaps
OBOMP	Oblique Optimized Matching Pursuit
OBOMPKSwaps	Oblique Optimized Matching Pursuit with k swaps
OMP	Orthogonal Matching Pursuit
OMPFinalRefi	Refinment of OMP
OMPKSwapRefi	Refiment of OOMP by kswapping and backward deleting steps
OMPKSwapping	extends OMPSwapping to considering K swaps
OMPKSwaps	Optimized Orthogonal Matching Pursuit with k swaps
OMPSwapping	Swapping based refinement of OMP method
OOMP	Optimized Orthogonal Matching Pursuit
OOMPFinalRefi	refinament of OOMP by swapping and backward deleting steps.
OOMPKSwapRefi	Refiment of OOMP by kswapping and backward deleting steps
OOMPKSwaps	Optimized Orthogonal Matching Pursuit with k swaps
Swapping	Swapping based refinement of OMP methods
VFSwapping	Swapping based refinement of OMP methods (inner product implementation)

1.2 Function-Description

1.2.1 BOOMP

Backward-Optimized Orthogonal Matching Pursuit

```
Using the Least Square criterion at each step it eliminates one function from a given
basis to have best possible representation in the reduced space. It also modifies the
corresponding biorthogonal functions.

Usage:    [ D, Di, beta, c ] = BOOMP( f, D, beta, tol, No );

Inputs:
  f       signal to represent
  D       set of chosen independent functions
  beta    set of biorthogonal functions to D
  tol     tolerance, desired difference between signal and its approx, (optional,
          default tol=1.0e-2)
```

```
  No      (optional) desired number of atoms in the decomposition if you want really No
          atoms set tol=0, it speeds the process

Outputs:
  D       new reduced set of independent functions
  Di      indices of atoms in new D w.r.t. to original D
  beta    biorthogonal functions to new D
  c       coefficients of the atomic decomposition

Note: this routine should be use only at the end of our selection process since it is
not adapting (for the speed purposes) unselected dictionary functions. Thus any of our
forward selection methods cannot be used after this.

References:
  M. Andrle, L. Rebollo-Neira, and E. Sagianos, "Backward-Optimized Orthogonal Matching
  Pursuit Approach", IEEE Signal Processing Letters, Vol (11,9), 705-708 (2004).
```

1.2.2 BOOMPQ

Backward-Optimized Orthogonal Matching and gives the orthonormal basis

```
Using the Least Square criterion at each step it eliminates one function from a given
basis to have best possible representation in the reduced space. It also modifies the
corresponding biorthogonal functions and orthonormal basis (obtained by modified
Gram-Schmidt).

Usage:    [ D, Di, Q, beta, c ] = BOOMPQ( f, D, Di, Q, beta, toli, No);
          [ D, Di, Q, beta, c ] = BOOMPQ( f, D, Di, Q, beta, tol );

Inputs:
  f       analyzing signal D dictionary D(:,1:N) is the selected basis, N=size(beta,2)
  Di      indices of atoms Q Q(:,1:N) orthonormal set spanning D(:,1:N), Q(:,N+1:end)
          the rest of the dictionary orthogonalized w.r.t Q(:,1:N)
  beta    set of biorthogonal functions to D(:,1:N) tol tolerance, difference between
          signal and approx, (optional, default tol=1.0e-2)
  No      (optional) desired number of atoms in the decomposition if you want really No
          atoms set tol=0, it speeds the process

Outputs:
  D       rearranged dictionary
          D(:,1:s) reduced basis, s=size(beta,2)
  Di      indices of atoms in new D w.r.t. to original D Q Q(:,1:s) orthonormal set
          spanning D(:,1:N), Q(:,s+1:end) the rest of the dictionary orthogonalized w.r.t
          Q(:,1:s)
  beta    biorthogonal functions to new D(:,1:s) c coefficients of the atomic
          decomposition

References:
  M. Andrle, L. Rebollo-Neira, and E. Sagianos, "Backward-Optimized Orthogonal Matching
  Pursuit Approach", IEEE Signal Processing Letters, Vol (11,9), 705-708 (2004).
```

1.2.3 KSwapping

extends Swapping to considering K swaps

```
Given an initial approximation of f, it improves upon the approximation by interchanging
swi-pairs of atoms (swi from the approximation and  swi from the dictionary) then
(swi+1)-atoms and (swi+1)-atoms, (swi+2)-atoms and (swi+2)-atoms and so on up to
```

```
sws-atoms, unless the desired precision tol has been reached. (See Ref[1]). If the
number of atoms involved in the swapping process is equal to sws and the stopping
criterion based on the precision tol has not been reached the function returns the value
re=0. Note: The inputs are obtainable by running OOMP first (see the example)

Usage:    [ re, resid, D, Di, beta, C, Q ] = KSwapping( f, D, Di, Q, beta, C, swi,...
              sws, tol );
          [ re, resid, D, Di, beta, C, Q ] = KSwapping( f, D, Di, Q, beta, C );

Inputs:
  f       signal to be decomposed
  D       dictionary, first k functions D(:,1:k) are the selected basis
  Di      indices of atoms in D with respect to the original dictionary
  beta    biorthogonal functions to D(:,1:k), k=size(beta,2)
  C       coefficients in the expansion
  Q       Q(:,1:k) orthonormal basis spanning the same space as D(:,1:k), Q(:,k+1:end)
          unselected atoms subtracted by their component in D(:,1:k)
  swi     minimum number of atoms to be swapped, default swi=1
  sws     maximum number of atoms to be swapped
  tol     tolerance for the approximation, default tol= 1e-8

Outputs:
  re      convergence indicator: re=1 if  the method converges within the given tol and
          re=0 otherwise
  resid   vector of length sws to store the residuals at each swapping. The first
          component of resid is the residual when swi atoms are swapped, the second
          component is the residual when (swi+1) atoms are swapped and so on. If the
          swapping is started from swi atoms, resid is of length sws-swi+1
  D       updated (re-arranged) dictionary, D(:,1:k) is the selected basis
  Di      indices of atoms in D with respect to the original dictionary
  beta    biorthogonal vectors to D(:,1:k)
  C       coefficients in the expansion
  Q       Q(:,1:k) orthonormal basis spanning the same space as D(:,1:k), Q(:,k+1:end)
          unselected atoms subtracted by their component in D(:,1:k)

References:
 [1] M. Andrle and L. Rebollo-Neira, "Improvement of Orthogonal Matching Pursuit
 strategies by Backward and Forward movements," in Proc. of the 31st International
 Conference on Acoustics, Speech, and Signal Processing (ICASSP'06)
 [2] M. Andrle and L. Rebollo-Neira, "A swapping-based refinement of orthogonal matching
 pursuit strategies", Signal Processing, Vol (86,3), 480-495 (2006)
 [3] L. Rebollo-Neira, "Measurements design and phenomena discrimination", J. Phys. A:
 Math. Theor. 42 (2009)

See also OOMPKSwaps OMPKSwaps OBOMPKSwaps Swapping OOMP
```

1.2.4 OBOMP

Oblique Optimized Matching Pursuit

```
Constructs an atomic decomposition which gives the oblique projection of a signal onto a
subspace of the span of V along span of WC.

It first generates orthogonal projections onto the orthogonal complement ofthe span of
WC and then find the atomic decomposition of the projected signal by the OOMP method.

For an example of how to use OBOMP to separate signal components run the code Exa_OBOMP
```

```
Usage:     [ fv ] = OBOMP( f, V, WC );
           [ fv, Vnew, Di, beta, c, U, Q ] = OBOMP( f, V, WC, err, No, opt, ind );

Inputs:
  f        signal to be projected
  V        dictionary for the space to project onto
  WC       dictionary spanning the space to project along
  err      error of each point of f (vector) or tolerance for the error's norm (scalar)
  No       (optional) maximal number of atoms to choose, if the number of chosen atoms
           equals to No, the routine will stop (default No=size(V,2))
  ind      (optional) indices determining  the initial subspace
  opt      (optional) chose the method for computing the orthogonal projector with
           OrthProj (default opt=2 with tolerance for linear independence 1e-7)

Outputs:
  Vnew     the dictionary V rearranged according to the selection process Vnew(:,1:k)
           contains the atoms chosen into the atomic decomposition
  Di       indices of atoms in Vnew written w.r.t the original V
  U        Dictionary for the orthogonal complement of WC (U(:,1:k)) is a basis
           bi-orthogonal to beta)
  Q        the first k columns Q(:,1:k) gives an orthonormal basis for the span of
           U(:,1:k)
  beta     'k' biorthogonal vectors to new  V(:,1:k) and U(:,1:k) (in span of U(:,1:k))
  c        'k' coefficients of the atomic decomposition for the oblique projection
  fv       oblique projection of f onto  new  V(:,1:k) along WC, i.e fv=Vnew(:,1:K)*c'

References
  [1] L. Rebollo-Neira, "Oblique Matching Pursuit", IEEE Signal Processing Letters,
  14,10, 703-707 (2007).
  [2] L. Rebollo-Neira, "Measurements design and phenomena discrimination", J. Phys. A:
  Math. Theor. 42 (2009).

See also  OBOMPKSwaps OOMP OOMPKSwaps
```

1.2.5 OBOMPKSwaps

Oblique Optimized Matching Pursuit with k swaps

```
Constructs an atomic decomposition which gives the oblique projection of a signal onto a
subspace of the span of V, along span of WC, using OOMPKSwaps.

It first takes the orthogonal projection onto the orthogonal complement of the span of
WC, then finds the atomic decomposition of the projected signal by the OOMP method and
corrects with KSwapping to obtain the sought projection.

for an example on how to use OBOMPKSwaps to separate signal components run the code
exa_OBOMPKSwaps

Usage:     [ re, resid, fv, Vnew, Di, beta, C, U, Q ] = OBOMPKSwaps( f, V, WC, err,...
               opt, No, ind, swi, sws, tols );
           [ re, resid, fv, Vnew, Di ] = OBOMPKSwaps( f, V, W );

Inputs:
  f        signal to be projected
  V        dictionary for the space to project onto
  WC       dictionary spanning the space to project along
```

```
  err     (optional) error of each point of f, or tolerance for the error's norm, before
          starting the corrections (default err=0.0001*norm(f))
  opt     (optional) chose the method for computing the orthogonal projector with
          OrthProj (default opt=2 with tolerance for linear Independence 1e-7)
  No      (optional) maximal number of atoms to choose,(default No=size(D,2))
  ind     (optional) indices determining the initial subspace
  swi     (optional) minimum number of atoms to be swapped, (default swi=1)
  sws     (optional) maximum number of atoms to be swapped, (default sws=size(beta,2))
  tols    (optional) tolerance for the  final approximation (default 0.0000001*norm(f))

Outputs:
  re      convergence indicator: re=1 if the method converges within the given tols and
          re=0 otherwise
  resid   vector of length sws to store the residuals at each swapping. The first
          component of resid is the residual when swi atoms are swapped, the second
          component  when (swi+1) atoms are swapped and so on.
  fv      oblique projection of f onto  new  V(:,1:k) along WC, i.e. fv=Vnew(:,1:K)*C'
  Vnew    the dictionary V rearranged according to the selection process Vnew(:,1:k)
          contains the atoms chosen to construct the atomic decomposition
  Di      indices of atoms in Vnew written w.r.t the original V
  beta    'k' biorthogonal vectors to new  V(:,1:k) and U(:,1:k)(spanning the same space
          as U(:,1:k)
  C       'k' coefficients of the atomic decomposition for the oblique projection
  U       Dictionary for the orthogonal complement of WC (U(:,1:k)) is a basis
          bi-orthogonal to beta)
  Q       the first k columns Q(:,1:k) gives an orthonormal basis for span of U(:,1:k)
          and beta.

References:
  [1] L. Rebollo-Neira, "Oblique Matching Pursuit", IEEE Signal Processing Letters,
  14,10, 703-707 (2007)
  [2] L. Rebollo-Neira, "Measurements design and phenomena discrimination", J. Phys. A:
  Math. Theor. 42 (2009)
```

1.2.6 OMP

Orthogonal Matching Pursuit

```
It creates an atomic decomposition of a signal using OMP criterion. You can choose a
tolerance, the number of atoms to take in or an initial subspace to influence the OMP
algorithm.


Usage:    [ Dnew, Di, beta, c ] = Omp( f, D, tol, No, ind );
          [ Dnew, Di ] = Omp( f, D, tol );

Inputs:
  f       analyzing signal
  D       dictionary of normalized atoms
  tol     desired distance between f and its approximation the routine will stop if
          norm(f'-Dsub*(f*beta)')*sqrt(delta)<tol where delta=1/L, L is number of points
          in a sample
  No      (optional) maximal number of atoms to choose, if the number of chosen atoms
          equals to No, routine will stop (default No=size(D,2)
  ind     (optional) indices determining  the initial subspace,

Outputs:
```

```
  D       the dictionary D rearranged according to the selection process D(:,1:k)
          contains the atoms chosen into the atomic decomposition
  Di      indices of atoms in new D written w.r.t the original D
  beta    'k' biorthogonal functions corresponding to new D(:,1:k)
  c       'k' coefficients of the atomic decomposition

References:
  L. Rebollo-Neira and D. Lowe, "Optimized Orthogonal Matching Pursuit Approach", IEEE
  Signal Processing Letters, Vol(9,4), 137-140, (2002).

See also OMPF.
```

1.2.7 OMPFinalRefi

Refinment of OMP

```
It creates an atomic decomposition for approximation a signal using OMP method up to a
given tolerance. When possible, the sparsity is improved afterwards by a combination of
swapping and backward deleting steps.

You can choose a  tolerance, the maximum number of atoms in the decomposition and an
initial subspace to influence the OOMP algorithm. Non-selected atoms subtracted by their
component in the chosen space are also available.

Usage:    [ D0, Di0 ] = OMPFinalRefi( f, D, tol );
          [ DS0, Di0, beta0, c0, Q0 ] = OMPFinalRefi( f, D, tol, No, ind);

Inputs:
  f       analyzing signal
  D       dictionary of normalized atoms
  tol     desired distance between f and its approximation the routine will stop if
          norm(f'-Dsub*(f*beta)')*sqrt(delta)<tol where delta=1/L (L is number of points
          in a sample) or delta=1, which is the default in OOMPF (to change this
          uncomment the corresponding line in OOMPF)
  No      (optional) maximal number of atoms to choose, if the number of chosen atoms
          equals to No, OOMP routine will stop (default No=size(D,2))
  ind     (optional) indices determining  the initial subspace for OOMP

Outputs:
  DS0     the dictionary D rearranged according to the selection process DS0(:,1:k)
          contains the atoms chosen into the atomic decomposition
  Di0     indices of atoms in DS0 written w.r.t the original D
  Q0      Q(:,1:k) contains orthonormal functions spanning  DS0(:,1:k) Q(:,k+1:N)
          contains  DS0(:,k+1:N) subtracted by the projection onto the space generated
          by  Q0(:,1:k)  (resp. DS0(:,1:k))
  beta0   'k' biorthogonal functions corresponding to new DS0(:,1:k)
  c0      'k' coefficients of the atomic decomposition
```

1.2.8 OMPKSwapRefi

Refiment of OOMP by kswapping and backward deleting steps

```
It creates an atomic decomposition for approximation a signal using OOMP method up to a
given tolerance. When possible, the sparsity is improved afterwards by a combination of
kswapping and backward deleting steps.

You can choose a tolerance, the maximum number of atoms in the decomposition and an
initial subspace to influence the OOMP algorithm. Non-selected atoms subtracted by their
```

```
component in the chosen space are also available.

Usage:    [ D0, Di0 ] = OMPKSwapRefi( f, D, tol );
          [ DS0, Di0, Q0, beta0, c0 ] = OMPKSwapRefi( f, D, tol, No, ind );

Inputs:
  f       analyzing signal
  D       dictionary of normalized atoms
  tol     desired distance between f and its approximation the routine will stop if
          norm(f'-Dsub*(f*beta)')*sqrt(delta)<tol where delta=1/L (L is number of points
          in a sample) or delta=1, which is the default in OOMPF (to change this
          uncomment the corresponding line in OOMPF)
  No      (optional) maximal number of atoms to choose, if the number of chosen atoms
          equals to No, OOMP routine will stop (default No=size(D,2)
  ind     (optional) indices determining  the initial subspace for OOMP
  swi     (optional) minimum number of atoms to swap (defaul=1)
  sws     (optional) maximum number of atoms to swap (defaul=all)
  []      can be used for sws, swi, ind, No, and tol

Outputs:
  DS0     the dictionary D rearranged according to the selection process DS0(:,1:k)
          contains the atoms chosen into the atomic decomposition
  Di0     indices of atoms in DS0 written w.r.t the original D
  beta0   'k' biorthogonal functions corresponding to new DS0(:,1:k)
  c0      'k' coefficients of the atomic decomposition
  Q0      Q(:,1:k) contains orthonormal functions spanning  DS0(:,1:k), Q(:,k+1:N)
          contains DS0(:,k+1:N) subtracted by the projection onto the space generated by
          Q0(:,1:k)  (resp. DS0(:,1:k))
```

1.2.9 OMPKSwapping

extends OMPSwapping to considering K swaps

```
Given an initial approximation of f, it improves upon the approximation by interchanging
swi-pairs of atoms (swi from the approximation and  swi from the dictionary) then
(swi+1)-atoms and (swi+1)-atoms, (swi+2)-atoms and (swi+2)-atoms and so on up to
sws-atoms, unless the desired precision tol has been reached. (See Ref[1]). If the
number of atoms involved in the swapping process is equal to sws and the stopping
criterion based on the precision tol has not been reached the function returns the value
re=0. Note: The inputs are obtainable by running OMP first (see the example)

For an example on how to use the routine see  exa_OOMPKSwaps

Usage:    [ re, resid, D, Di, beta, C, Q ] = OMPKSwapping( f, D, Di, Q, beta, C, swi,...
              sws, tol );
          [ re, resid, D, Di, beta, C, Q ] = OMPKSwapping( f, D, Di, Q, beta, C );

Inputs:
  f       analysing signal
  D       dictionary, first k functions D(:,1:k) are the selected basis
  Di      indices of atoms in D with respect to the original dictionary
  Q       Q(:,1:k) orthonormal basis spanning the same space as D(:,1:k)
          Q(:,k+1:end) unselected atoms subtracted by their component in D(:,1:k)
  beta    biorthogonal functions to D(:,1:k), k=size(beta,2)
  C       coefficients in the expansion
  swi     minimum number of atoms to be swapped, default swi=1
  sws     maximum number of atoms to be swapped
```

```
  tol      tolerance for the approximation, default tol= 1e-8

 Outputs:
  re       convergence indicator: re=1 if  the method converges within the given tol and
           re=0 otherwise
  resid    vector of length sws to store the residuals at each swapping. The first
           component of resid is the residual when swi atoms are swapped, the second
           component is the residual when (swi+1) atoms are swapped and so on. If the
           swapping is started from swi atoms, resid is of length sws-swi+1
  D        updated (re-arranged) dictionary, D(:,1:k) is the selected basis
  Di       indices of atoms in D with respect to the original dictionary
  beta     biorthogonal vectors to D(:,1:k)
  C        coefficients in the expansion
  Q        Q(:,1:k) orthonormal basis spanning the same space as D(:,1:k)
           Q(:,k+1:end) unselected atoms subtracted by their component in D(:,1:k)

 References:
  [1] M. Andrle and L. Rebollo-Neira, "Improvement of Orthogonal Matching Pursuit
  strategies by Backward and Forward movements," in Proc. of the 31st International
  Conference on Acoustics, Speech, and Signal Processing (ICASSP'06)
  [2] M. Andrle and L. Rebollo-Neira, "A swapping-based refinement of orthogonal matching
  pursuit strategies", Signal Processing, Vol (86,3), 480-495 (2006)
  [3] L. Rebollo-Neira, "Measurements design and phenomena discrimination",%   J. Phys.
  A: Math. Theor. 42 (2009)

 See also OOMPKSwaps OMPKSwaps OBOMPKSwaps Swapping OOMP BOOMP OBOMP
```

1.2.10 OMPKSwaps

Optimized Orthogonal Matching Pursuit with k swaps

```
 Constructs an approximation of f using OOMP and improves the approximation with
 KSwapping interchanging swi-pairs of atoms (swi from the approximation and swi from the
 dictionary) then (swi+1)-atoms and (swi+1)-atoms  and so on up to sws-atoms, if tols is
 not reached. If the stopping criterion based on the precision tols  is not reached re=0
 is returned. (See KSwapping)

 Usage:    [ re, resid, D, Di, beta, C, Q ] = OOMPKSwaps( f, D );
           [ re, resid, D, Di, beta, C, Q ] = OOMPKSwaps( f, D, err, No, ind, swi,...
               sws, tols );

 Inputs:
  f        signal to be represenated
  D        dictionary for the space to project onto
  err      (optional) error of each point of f, or tolerance for the error's norm, before
           starting the swappings (default err=0.0001*norm(f))
  No       (optional) maximal number of atoms to choose,(default No=size(D,2))
  ind      (optional) indices determining  the initial subspace
  swi      (optional) minimum number of atoms to be swapped, (default swi=1)
  sws      (optional) maximum number of atoms to be swapped, (default sws=size(beta,2))
  tols     (optional) tolerance for the final approximation (default 0.0000001*norm(f))

Outputs:
  re       convergence indicator: re=1 if the method converges within the given tols and
           re=0 otherwise
  resid    vector of length sws to store the residuals at each swapping. The first
           component of resid is the residual when swi atoms are swapped, the second
```

```
          component  when (swi+1) atoms are swapped and so on.
  D       updated (re-arranged) dictionary, D(:,1:k) is the selected basis
  Di      indices of atoms in D with respect to the original dictionary
  beta    biorthogonal vectors to D(:,1:k)
  C       coefficients in the expansion
  Q       Q(:,1:k) orthonormal basis spanning the same space as D(:,1:k)
          Q(:,k+1:end) unselected atoms subtracted by their component in D(:,1:k)

References:
 [1] M. Andrle and L. Rebollo-Neira, "Improvement of Orthogonal Matching Pursuit
 strategies by Backward and Forward movements," in Proc. of the 31st International
 Conference on Acoustics, Speech, and Signal Processing (ICASSP'06)
 [2] M. Andrle and L. Rebollo-Neira, "A swapping-based refinement of orthogonal matching
 pursuit strategies", Signal Processing, Vol (86,3), 480-495 (2006)
 [3] L. Rebollo-Neira, "Measurements design and phenomena discrimination", J. Phys. A:
 Math. Theor. 42 (2009)
 [4] L. Rebollo-Neira and D. Lowe, "Optimized Orthogonal Matching Pursuit Approach", IEEE
 Signal Processing Letters, Vol(9,4), 137-140, (2002).
```

1.2.11 OMPSwapping

Swapping based refinement of OMP method

```
It interchanges at each step one atom from the atomic decomposition with another atom
from the dictionary to improve the OMP approximation via adaptive biorthogonalization.
At each step it modifies the biorthogonal vectors giving rise to the duals of selected
atoms. The inputs are obtainable from the outputs of  the OMP function

Usage:    [ D, Di, Q, beta ] = OMPSwapping( f, D, Di, Q, beta );

Inputs:
  D       dictionary, first k functions D(:,1:k) are the selected basis
  Di      indices of atoms in D with respect to the original dictionary
  beta    biorthogonal vectors to D(:,1:k), k=size(beta,2)
  Q       orthonormal basis spanning the same space as D(:,1:k)

Outputs:
  D       updated (re-arranged) dictionary, D(:,1:k) are the selected basis
  beta    biorthogonal functions to D(:,1:k), k=size(beta,2)
  Di      indices of atoms in D with respect to the original dictionary
  Q       Q orthonormal basis spanning the same space as D(:,1:k)

References:
  M. Andrle and L. Rebollo-Neira, "A swapping-based refinement of orthogonal matching
  pursuit strategies", Signal Processing, Vol 86, No 3, pp. 480-495, 2006.

See also OOMPSwapping, OOMPKSwaps, OMPKSwaps, OMPKSwapping
```

1.2.12 OOMP

Optimized Orthogonal Matching Pursuit

```
It creates an atomic decomposition of a signal using OOMP method [1]. You can choose a
tolerance, the number of atoms to take in or an initial subspace to influence the OOMP
algorithm. Non-selected atoms subtracted by their component in the chosen space are also
available.

Usage:    [ Dnew, beta, Di ] = OOMP( f, D, tol );
```

```
         [ Dnew, beta, Di, c, Q ] = OOMP( f, D, tol, No, ind );

Inputs:
  f       signal to be represented
  D       dictionary of atoms
  tol     desired distance between f and its approximation the routine will stop if
          norm(f'-Dsub*(f*beta)')*sqrt(delta)<tol where delta=1/L, L is number of points
          in a sample
  No      (optional) maximal number of atoms to choose, if the number of chosen atoms
          equals to No, routine will stop (default No=size(D,2))
  ind     (optional) indices determining  the initial subspace,

Outputs:
  D       the dictionary D rearranged according to the selection process D(:,1:k)
          contains the atoms chosen into the atomic decomposition
  beta    'k' biorthogonal functions corresponding to new D(:,1:k)
  Di      indices of atoms in new D written w.r.t the original D
  c       'k' coefficients of the atomic decomposition
  Q       Q(:,1:k) contains orthonormal functions spanning new D(:,1:k), Q(:,k+1:N)
          contains new D(:,k+1:N) subtracted by the projection onto the space generated
          by  Q(:,1:k)  (resp. D(:,1:k))

References:
  [1] L. Rebollo-Neira and D. Lowe, "Optimized Orthogonal Matching Pursuit Approach",
  IEEE Signal Processing Letters, Vol(9,4), 137-140, (2002).
  For the current implementation:
  [2] M. Andrle and L. Rebollo-Neira, "A swapping-based refinement of orthogonal
  matching pursuit strategies", Signal Processing, Vol 86, No 3, pp. 480-495, (2006).

See also OMP Swapping OOMPKSwaps OMPKSwaps OOMPFinalRefi BOOMP OBOMP
```

1.2.13 OOMPFinalRefi

refinament of OOMP by swapping and backward deleting steps.

```
It creates an atomic decomposition for approximation a signal using OOMP method up to a
given tolerance. When possible, the sparsity is improved afterwards by a combination of
swapping and backward deleting steps.

You can choose a  tolerance, the maximum number of atoms in the decomposition and an
initial subspace to influence the OOMP algorithm. Non-selected atoms subtracted by their
component in the chosen space are also available.

Usage:   [ D0, Di0 ] = OOMPFinalRefi( f, D, tol );
         [ DS0, Di0, beta0, c0,Q0 ] = OOMPFinalRefi( f, D, tol, No, ind);

Inputs:
  f       analyzing signal
  D       dictionary of normalized atoms
  tol     desired distance between f and its approximation the routine will stop if
          norm(f'-Dsub*(f*beta)')*sqrt(delta)<tol where delta=1/L (L is number of points
          in a sample) or delta=1, which is the default in OOMP (to change this
          uncomment the corresponding line in OOMP)
  No      (optional) maximal number of atoms to choose, if the number of chosen atoms
          equals to No, OOMP routine will stop (default No=size(D,2)
  ind     (optional) indices determining  the initial subspace for OOMP
  []      can be used for ind, No tol
```

```
Outputs:
  DS0     the dictionary D rearranged according to the selection process DS0(:,1:k)
          contains the atoms chosen into the atomic decomposition
  Di0     indices of atoms in DS0 written w.r.t the original D
  Q0      Q(:,1:k) contains orthonormal functions spanning  DS0(:,1:k), Q(:,k+1:N)
          contains  DS0(:,k+1:N) subtracted by the projection onto the space generated
          by Q0(:,1:k)  (resp. DS0(:,1:k))
  beta0   'k' biorthogonal functions corresponding to new DS0(:,1:k)
  c0      'k' coefficients of the atomic decomposition
```

1.2.14 OOMPKSwapRefi

Refiment of OOMP by kswapping and backward deleting steps

```
It creates an atomic decomposition for approximation a signal using OOMP method up to a
given tolerance. When possible, the sparsity is improved afterwards by a combination of
kswapping and backward deleting steps.

You can choose a tolerance, the maximum number of atoms in the decomposition and an
initial subspace to influence the OOMP algorithm. Non-selected atoms subtracted by their
component in the chosen space are also available.

Usage:    [ D0, Di0 ] = OOMPKSwapRefi( f, D, tol );
          [ DS0, Di0, Q0, beta0, c0 ] = OOMPKSwapRefi( f, D, tol, No, ind );

Inputs:
  f       analyzing signal
  D       dictionary of normalized atoms
  tol     desired distance between f and its approximation the routine will stop if
          norm(f'-Dsub*(f*beta)')*sqrt(delta)<tol where delta=1/L (L is number of points
          in a sample) or delta=1, which is the default in OOMPF (to change this
          uncomment the corresponding line in OOMPF)
  No      (optional) maximal number of atoms to choose, if the number of chosen atoms
          equals to No, OOMP routine will stop (default No=size(D,2)
  ind     (optional) indices determining  the initial subspace for OOMP
  swi     (optional) minimum number of atoms to swap (defaul=1)
  sws     (optional) maximum number of atoms to swap (defaul=all)
  []      can be used for sws, swi, ind, No, and tol

Outputs:
  DS0     the dictionary D rearranged according to the selection process DS0(:,1:k)
          contains the atoms chosen into the atomic decomposition
  Di0     indices of atoms in DS0 written w.r.t the original D
  beta0   'k' biorthogonal functions corresponding to new DS0(:,1:k)
  c0      'k' coefficients of the atomic decomposition
  Q0      Q(:,1:k) contains orthonormal functions spanning  DS0(:,1:k), Q(:,k+1:N)
          contains DS0(:,k+1:N) subtracted by the projection onto the space generated by
          Q0(:,1:k) (resp. DS0(:,1:k))
```

1.2.15 OOMPKSwaps

Optimized Orthogonal Matching Pursuit with k swaps

```
Constructs an approximation of f using OOMP and improves the approximation with
KSwapping interchanging swi-pairs of atoms (swi from the approximation and swi from the
dictionary)  then (swi+1)-atoms and (swi+1)-atoms  and so on up to sws-atoms, if tols is
not reached. If the stopping criterion based on the precision tols  is not reached re=0
```

```
is returned. (See KSwapping)

For an example on how to use OOMPKSwaps to represent a signal run de code exa_OOMPKSwaps


Usage:     [ re, resid, D, Di, beta, C, Q ] = OOMPKSwaps( f, D );
           [ re, resid, D, Di, beta, C, Q ] = OOMPKSwaps( f, D, err, No, ind, swi,...
               sws, tols );

Inputs:
  f        signal to be represenated
  D        dictionary for the space to project onto
  err      (optional) error of each point of f, or tolerance for the error's norm, before
           starting the swappings (default err=0.0001*norm(f))
  No       (optional) maximal number of atoms to choose,(default No=size(D,2))
  ind      (optional) indices determining  the initial subspace
  swi      (optional) minimum number of atoms to be swapped, (default swi=1)
  sws      (optional) maximum number of atoms to be swapped, (default sws=size(beta,2))
  tols     (optional) tolerance for the final approximation (default 0.0000001*norm(f))

Outputs:
  re       convergence indicator: re=1 if the method converges within the given tols and
           re=0 otherwise
  resid    vector of length sws to store the residuals at each swapping. The first
           component of resid is the residual when swi atoms are swapped, the second
           component  when (swi+1) atoms are swapped and so on.
  D        updated (re-arranged) dictionary, D(:,1:k) is the selected basis
  Di       indices of atoms in D with respect to the original dictionary
  beta     biorthogonal vectors to D(:,1:k)
  C        coefficients in the expansion
  Q        Q(:,1:k) orthonormal basis spanning the same space as D(:,1:k), Q(:,k+1:end)
           unselected atoms subtracted by their component in D(:,1:k)

References:
 [1] M. Andrle and L. Rebollo-Neira, "Improvement of Orthogonal Matching Pursuit
 strategies by Backward and Forward movements," in Proc. of the 31st International
 Conference on Acoustics, Speech, and Signal Processing (ICASSP'06)
 [2] M. Andrle and L. Rebollo-Neira, "A swapping-based refinement of orthogonal matching
 pursuit strategies", Signal Processing, Vol (86,3), 480-495 (2006)
 [3] L. Rebollo-Neira, "Measurements design and phenomena discrimination",%   J. Phys.
 A: Math. Theor. 42 (2009)
```

1.2.16 Swapping

Swapping based refinement of OMP methods

```
It interchange at each step one atom from the atomic decomposition for another atom from
the dictionary to improve the signal approximation. Similarly at each step it modifies
the biorthogonal vectors and the unselected atoms from the dictionary subtracted by
their component from the selected space. The process is carried out until the
approximation error will not increase.

Usage:     [ D, Di, Q, beta ] = Swapping( f, D, Di, Q, beta );

Inputs:
  D        dictionary, first k functions D(:,1:k) are the selected basis
  Di       indices of atoms in D with respect to the original dictionary
```

```
  beta    biorthogonal functions to D(:,1:k), k=size(beta,2)
  Q       Q(:,1:k) orthonormal basis spanning the same space as D(:,1:k)
          Q(:,k+1:end) unselected atoms subtracted by their component in D(:,1:k)

Outputs:
  D       updated (re-arranged) dictionary, D(:,1:k) are the selected basis
  beta    biorthogonal functions to D(:,1:k), k=size(beta,2)
  Di      indices of atoms in D with respect to the original dictionary
  Q       Q(:,1:k) orthonormal basis spanning the same space as D(:,1:k)
          Q(:,k+1:end) unselected atoms subtracted by their component in D(:,1:k)

References:
  M. Andrle and L. Rebollo-Neira, "A swapping-based refinement of orthogonal matching
  pursuit strategies", Signal Processing, Vol 86, No 3, pp. 480-495, 2006.

See also KSwapping, VFSwapping.
```

1.2.17 VFSwapping

Swapping based refinement of OMP methods (inner product implementation)

```
It interchange at each step one atom from the atomic decomposition for another atom from
the dictionary to improve the signal approximation.  The process is carried out until
the approximation error will not increase. The routine is not modifying biorthogonal
functions or unselected atoms subtracted by the component in the selected space but only
several related inner product matrices. See Alg. 4b in the reference below.

Usage:    VFSwapping( f, D );

Inputs:
 f        signal
 D        dictionary

These inputs and all outputs are realized through global variables AL LL LF J IJ I
  I       set of all indices
  J       set of indices (subset of I) which are not in the basis
  IJ      set of indices (subset of I) which are in the basis
  AL      inner products between dictionary D and lambdas
  LL      inner products between lambda(IJ), between biorthogonal functions to D(:,IJ)
  LF      inner products between lambda and signal f

Note:     lambda(IJ) means biorthogonal functions to D(IJ)
          lambda(J) means D(J) subtracted by their component in selected basis D(IJ)

References:
  M. Andrle and L. Rebollo-Neira, "A swapping-based refinement of orthogonal matching
  pursuit strategies", Signal Processing, Vol (86,3), 480-495 (2006).

See also Swapping, KSwapping.
```

Chapter 2

Lq Minimization

2.1 Function-Summary

ALqMin	adaptive lq like minimization
LqFOCUSS	solves for x in $f = Dx$ by minimizing the lq^q norm by the method FOCUSS
RegFOCUSS	solves for xc in $f = Dxc$ by minimizing $\|\|xc\|\|_q^q + lam(f - Dxc)$

2.2 Function-Description

2.2.1 ALqMin

adaptive lq like minimization

```
It gives an approximated solution to an underdetermined least square problem by
minimzation of the lq^q-like quatity. The algorithm evolves by adaptive selection of a
subset of normal equations as linear constraints for the minimization process [1].  The
constrained minimization is implemented by the function LQ_FOCUSS, which applies the
FOCUSS algorithm [2].

Usage:    [ xc ] = ALqMin( fw, U, q );
          [ xc, ki, resn ] = ALqMin( fw, U, q, tol, No, itmax );

Inputs:
  fw      data to be modeled  by fw~U*xc
  U       matrix in the model above
  q       specifies the value q for the lq norm
  tol     the routine will stop if norm(fw-U*xc)<tol  (default tol=1e-8)
  No      maximum number of constraints (stopping condition) (default No=size(U,2))
  itmax   maximal number of iterations for the focuss algorithsm (default itmax=30)

Outputs:
  xc      solution of the lq minimization
  Nc      total number of normal equations that have been considered
  resn    value of norm(fw-U*xc)*sqrt(delta)

References:
 [1] L. Rebollo-Neira and A. Plastino, Nonlinear non-extensive approach for
 identification of structured information,  Physica A, 2009
 [2] B.D. Rao and K. Kreutz-Delgado,  An Affine Scaling Methodology for Best Basis
 Selection, IEEE Trans. Sig. Proc. 47 (1999) 187--200
```

2.2.2 LqFOCUSS

solves for x in $f = Dx$ by minimizing the lq^q norm by the method FOCUSS

```
Usage:    [ xc ] = LqFOCUSS( f, D, xo, q )
          [ xc ] = LqFOCUSS( f, D, xo, q, tol, itmax )

Inputs:
  f       data  modelled as f= D xc
  D       matrix in the model
  xo      initial solution
  q       value for q in the lq norm like measure
  tol     tolerance for convergence (default tol = 1e-8)
  itmax   maximum number of iterations (default itmax=30)

Outputs:
   xc     solution of minimum lq norm in the model f= D*xc

Reference
  [1] B.D. Rao and K. Kreutz-Delgado,  An Affine Scaling Methodology for Best Basis
  Selection, IEEE Trans. Sig. Proc. 47 (1999) 187--200
```

2.2.3 RegFOCUSS

solves for xc in $f = Dxc$ by minimizing $||xc||_q^q + lam(f - Dxc)$

```
It applies  the algorithm given in FOCUSS [1]

Usage:    [xc] = RegFOCUSS( f, D, q )
          [xc] = RegFOCUSS( f, D, q, lam, tol, itmax, xo )

Inputs:
  f      data  modelled as f= D xc
  D      matrix in the model
  q      value for q in the lq norm like measure
  lam    regularization parameter (default lam = 1e-8)
  tol    tolerance for convergence (default tol = 1e-8)
  itmax  maximum number of iterations (default itmax=30)
  xo     initial solution (default all the entris equal to 1)
  []     can be used for itmax, tol and lam

Outputs
  xo     regularized solution of f= D xc

References
  [1] B.D. Rao, K. Engan, S. F. Cotter,  J. Palmer, K. Kreutz-Delgado, Subset selection
  in noise based on diversity measure minimization, IEEE Transactions on Signal
  Processing, 51, 3  (2003) 760-- 770, 10.1109/TSP.2002.808076.
```

Chapter 3

Examples

3.1 Example-Summary

exa_OBOMP	using the OBOMP function
exa_OOMPKSwaps	improves upon the OOMP approximation by k swappings
exa_chirp	adapted spline approximation of the chirp signal

3.2 Example-Description

3.2.1 exa_OBOMP

using the OBOMP function

```
 It separates the component fv in V  from f=fv+fw; with fw in WC (uses OBOMP)
```

3.2.2 exa_OOMPKSwaps

improves upon the OOMP approximation by k swappings

```
Creates a cardinal B-spline dictionary and a the signal f as a random superposition of
95 splines. Calls OOMP_KSWAPS (which uses KSwapping) to improve the OOMP approximation
by k swappings, untill tols=1e-9*norm(f) is reached.
```

3.2.3 exa_chirp

adapted spline approximation of the chirp signal

```
This example generates a nonuniform spline space adapted to a chirp signal and
constructs a dictionary for sparse approximation of the chirp through refinements of
OOMP and OMP approaches.
```

Part II

Projectors and Duals

Chapter 4

Projectors

4.1 Function-Summary

ObliProj	constructs an oblique projection matrix onto the span of the columns of V along the span of the columns of Wperp
OptObliProj	regularizes the oblique projector of a give signal by truncation of singular values
OrthProj	construct an orthogonal projection matrix onto the span of the columns of D.
RegObliProj	regularizes the oblique projection of a given noisy signal by truncation of singular valueas

4.2 Function-Description

4.2.1 ObliProj

constructs an oblique projection matrix onto the span of the columns of V along the span of the columns of Wperp

```
for opt=[1,1] computes the projector as E=V*W, where W=pinv(U'*V)*U'
         with U=V-orth_proj_{Wperp} V

for opt=[2,1] computes the projector as E=V*W, where W=pinv(U'*U)*U'
         with U=V-OrthProj_{Wperp} V

for opt=[3,1] computes the projector as E=V*W, where W=pinv(U)'
         with U=V-OrthProj_{Wperp} V

for opt=[1,2] computes the projector as E=V*W, where W=pinv(Q'*V)*Q'
         with  U=V-OrthProj_{Wperp}  and Q=DREOr(U)

for opt=[1,3] computes the projector as E=V*W, where W=pinv(Q'*V)*Q'
         with  U=V-OrthProj_{Wperp}  and Q=qr(U)

for opt=[1,4] computes the projector as E=V*W, where W=pinv(Q'*V)*Q'
         with  U=V-OrthProj_{Wperp}  and Q=orth(U)

for opt=[2,2] computes the projector as E=V*W, where W=pinv(Q'*U)*Q'
         with  U=V-OrthProj_{Wperp}  and Q=DREOr(U)

for opt=[2,3] computes the projector as E=V*W', where W'=pinv(Q'*U)*Q'
         with  U=V-OrthProj_{Wperp}  and Q=qr(U)
```

```
for opt=[2,4] computes the projector as E=V*W', where W'=pinv(Q'*U)*Q'
         with  U=V-OrthProj_{Wperp}  and Q=orth(U)

Usage:    [ W, U, E ] = ObliProj( V, Wperp, opt, tol, ind );
          [ W, U, E ] = ObliProj( V, Wperp );

Inputs:
  V       matrix the columns of which span then space to project onto
  Wperp   matrix the columns of which span then space to project along
  opt     array to chose the method to calculate the projector (see above) default
          opt=[3,1]
  tol     if opt=[1,2], or [2,2]  tol is the torance set for considering linearly
          independent columns [default tol= 1.0000e-7]
  ind     (optional) the indices of vectors to start the orthogonalization (see DREOr)
  []      can be used for ind, tol and opt

Outputs:
  W       matrix producing E=U*W;
  U       matrix producing E=U*W;
  E       Projector onto span of columns of V onto columns of Wperp

see orth DREOrp DREOr qr
```

4.2.2 OptObliProj

regularizes the oblique projector of a give signal by truncation of singular values

```
The projection of f onto span of V  along span of WC is regularized by truncation of
singular values- the number of singular values is decided by minimizing:

                 ||P_W f' -  P_W E_{VWC}f'||

where P_W is the orthogonal projector onto W (the orthogonal complement of span WC) and
E_{V,WC}) is the oblique projector onto span V along span WC.

Usage:    [ fe, c, lm ] = OptObliProj( D, WC, f, opt, eps );
          [ fe, c, lm ] = OptObliProj( D, WC, f );
Inputs:
  V       matrix, the columns of which span the space to project onto
  WC      matrix, the columns of which span the space to project along
  f       vector, with signal to be projected
  opt     equivalent role as in ObliProj and (for details see there) but some of them
          may not always be good here default opt=[3,1].
  eps     minimum eigenvalue to be considered nonzero
  []      can be used for eps or/and opt

Outputs:
  fe      regularized oblique projection of f
  c       coefficients in the decomposition fe=V*c
  lm      resulting number of nonzero eigenvalues to calculate fe

See  also ObliProj RegObliProj OrthProj
```

4.2.3 OrthProj

construct an orthogonal projection matrix onto the span of the columns of D.

```
for opt=1 it uses the matlab function orth to orthogonalize D
for opt=2 it uses the routine DREOrp (recommended when the columns of D are quasi
          Linearly Dependent up tolerance tol, see DREor)
for opt=3 it uses the QR decomposition matlab function.

Usage:    [ P, Q ] = OrthProj( D, opt, tol, ind );
          [ P, Q ] = OrthProj( D );

Inputs:
  D       matrix the columns of which is a dictionary of normalized atoms
  opt     to chose the method to calculate the projector (see above) default for opt=3
  tol     (optional) the tolerance set for considering linearly dependant columns
          [default tol = 1.0000e-7]
  ind     (optional) the indices of vectors to start the orthogonalization (see DreOr)

Outputs:
  Q       Orthogonal vectors such that P=Q*Q'
  P       Projector onto span of the columns of Q

See also DREOrp DREOr qr orth
```

4.2.4 RegObliProj

regularizes the oblique projection of a given noisy signal by truncation of singular valueas

```
The regularization tries to fulfill:

||P_W f' -  P_W E_{VWC}f'||\le ||  P_W err'||,

where P_W is the orthogonal projector onto W (the orthogonal complement of span WC) and
E_{V,WC} is the oblique projector onto span V along span WC.

Usage:    [ fe, c, lm ] = RegObliProj( D, WC, f, err, opt, No );
          [ fe, c, lm ] = RegObliProj( D, WC, f );

Inputs:
  V       matrix, the columns of which span the space to project onto
  WC      matrix, the columns of which span the space to project along
  f       vector, with signal to be projected
  opt     equivalent role as in ObliProj (for details see there) default opt=[1,2] (with
          tol_orth= 1e-7 for DREOr)
  err     (optional) error of each point of f, or tolerance for the error's norm default
          err=ones(size(f))*5e-7
  No      maximum number of eigenvalues to consider default all
  []      can be used for err, opt, or No

Outputs:
fe      regularized oblique projection of f
  c       coefficients in the decomposition fe=V*c
  lm      resulting number of nonzero eigenvalues to calculate fe
```

Chapter 5

Duals

5.1 Function-Summary

BioBack	deletes the requested vector j from a given basis taken from a dictionary
BioDictDel	deletes a vector from a basis selected from a given dictionary and appropriately modifies biorthogonal functions, orthonormal functions spanning the same space as the basis, and unselected dictionary atoms subtracted by their components in the selected basis.
BioDictIns	enlarges a basis selected from a given dictionary by one vector
BioFor	enlargers the dual/biorthogonal basis enlarger by one vector
BioInsert	adds an atom to a basis. It also appropriately modifies the corresponding biorthogonal basis and orthonormal basis (obtained by modified Gram-Schmidt).
DRE	Dictionary Redundancy Elimination
DREOr	uses DRE method to produce and orthogonal basis from a dictionary and gives dictionary's indices of the atoms spannig the space.
DREOrp	uses Dre method to produce and orthogonal basis from a dictionary.
FrDelete	deletes the requested vector from a given frame
FrInsert	adds a vector to a frame. It also appropriately modifies the corresponding dual frame.
FrInsertBlock	adds vectors to a basis and gives the duals spanning the same space
NBioDictIns	enlarges a basis selected from a given dictionary by one vector. It modifies the biorthogonal basis for the same space and gives the projection of the remainding atoms onto the orthogonal complementary space
NBioInsert	adds an atom to a basis and modifies the biorthogonal basis for the same space

5.2 Function-Description

5.2.1 BioBack

deletes the requested vector j from a given basis taken from a dictionary

```
Modifies the corresponding biorthogonal basis and orthonormal basis (obtained by
modified Gram-Schmidt) and retuned the re-ordered dictionary (this is the only diference
with biodelete, which does not deal with the whole dictionary).

Usage:    [ D, psin, beta ] = BioBack( D, psin, beta, j );

Inputs:
  D       Dictionary, first kb vectors D(:,1:kb) are the selected basis
```

```
  psin    orthonormal basis spanning the same space as D(:,1:kb) kb=size(psin,2
  beta    biorthogonal basis to D(:,1:kb),  kb=size(beta,2)
  j       index of function to eliminate

Outputs:
  D       updated (rearranged) dictionary (D(:,1:kb-1) is the new basis)
  psin    updated orthonormal basis spanning the same space as D(:,1:kb-1)
  beta    updated biorthogonal basis to D(:,1:kb-1)

References:
  L. Rebollo-Neira, "Recursive bi-orthogonalisation approach and orthogonal projectors",
  math-ph/0209026 (2002).

See also NBioDelete, NBioDictDel, BioDictDel.
```

5.2.2 BioDictDel

deletes a vector from a basis selected from a given dictionary and appropriately modifies biorthogonal functions, orthonormal functions spanning the same space as the basis, and unselected dictionary atoms subtracted by their components in the selected basis.

```
The dictionary is then rearranged to have one to one correspondence with  beta (dual
functions) and psin.

Usage:  [ D, psin, beta ] = BioDictDel( D, psin, beta, j );

Inputs:
  D       dictionary, first kb functions are the selected basis
  psin    psin(:,1:kb) orthonormal functions spanning the same space as D(:,1:kb)
          psin(:,kb+1:end)=D(:,kb+1:end) without component from D(:,1:kb)
  beta    kb biorthogonal functions to D(:,1:kb), kb=size(beta,2)
  j       index of function to eliminate

Outputs:
  D       updated (rearranged) dictionary
  psin    psin(:,1:kb-1) updated (recalculated) orthonormal functions spanning the same
          space as the chosen atoms psin(:,kb:end) updated (recalculated) unchosen
          dictionary atoms subtracted by their components in the selected basis.
  beta    updated biorthogonal functions to D(:,1:kb-1)

References:
   L. Rebollo-Neira, "Recursive bi-orthogonalisation approach and orthogonal projectors",
   math-ph/0209026 (2002).

See also: NBioDictDel, BioDelete, NBioDelete.
```

5.2.3 BioDictIns

enlarges a basis selected from a given dictionary by one vector

```
It appropriately modifies the biorthogonal functions and unselected dictionary atoms
subtracted by their components in the selected basis. It also updates the set of
orthonormal functions spanning the same space as chosen atoms.

Usage:     [ Q, beta ] = BioDictIns( D, Q, beta, k );

Inputs:
  D       dictionary, rearranged in such way that D(:,1:k-1) are the selected atoms
```

```
          D(:,k) is the atom to add into the basis
  Q       Q(:,1:k-1)= orthonormal basis to D(:,1:k-1) Q(:,k:end)=D(:,k:end) without
          component from D(:,1:k-1)
  beta    biorthogonal functions to D(:,1:k-1)
  k       k-th atom of D to incorporate it says that first k-1 atoms are already in
          basis

Outputs:
 Q        Q(:,1:k)= orthonormal basis to D(:,1:k) Q(:,k+1:end)=D(:,k+1:end) without
          component from D(:,1:k)
 beta     updated biorthogonal functions to D(:,1:k)

References:
  L. Rebollo-Neira, "Recursive bi-orthogonalisation approach and orthogonal projectors",
  math-ph/0209026 (2002).

See also NBioDictIns, BioInsert, NBioInsert.
```

5.2.4 BioFor

enlargers the dual/biorthogonal basis enlarger by one vector

```
It modifies the biorthogonal basis  and updates the set of orthonormal vectors spanning
the same space as the the enlaged basis.

Usage:     [ Q, beta ] = BioFor( D, Q, beta, k );

Inputs:
  D       dictionary, rearranged in such way that D(:,1:k-1) are the basis vectors and
          D(:,k) is the atom to add into the basis
  Q       Q(:,1:k-1)= orthonormal basis for the span of D(:,1:k-1)
  beta    biorthogonal basis to D(:,1:k-1)
  k       k-th element of D to incorporate in the basis it implies that the first k-1
          vectors are already in the basis

Outputs:
  Q       Q(:,1:k)= orthonormal basis  for span of  D(:,1:k)
  beta    updated biorthogonal basis to D(:,1:k)

References:
  L. Rebollo-Neira, "Recursive bi-orthogonalisation approach and orthogonal projectors",
  math-ph/0209026 (2002).

See also NBioDictIns, BioInsert, NBioInsert
```

5.2.5 BioInsert

adds an atom to a basis. It also appropriately modifies the corresponding biorthogonal basis and orthonormal basis (obtained by modified Gram-Schmidt).

```
Usage:    [ D, psin, beta ] = BioInsert( D, psin, beta, atom, tol );

Inputs:
  D       already selected basis
  psin    orthonormal basis spanning the same space as D
  beta    biorthogonal basis to D
  atom    new atom to be incorporated into basis
  tol     tolerance (optional parameter) to decide linear dependence default value =
```

```
          1.0000e-7

 Outputs:
   D       updated basis
   psin    updated orthonormal basis spanning the same space as D
   beta    updated biorthogonal functions to D

 References:
   L. Rebollo-Neira, "Recursive bi-orthogonalisation approach and orthogonal projectors",
   math-ph/0209026 (2002).

 See also NBioInsert, NBioDictIns, BioDictIns.
```

5.2.6 DRE

Dictionary Redundancy Elimination

```
 With help of column pivoting it tries  to choose  a stable basis from a given
 dictionary.

 Usage:     [ Dnew, Di, Q, beta ] = DRE( D, tol, ind );
            [ Dnew, Di ] = DRE( D );

 Inputs:
   D        dictionary of normalized atoms
   tol      tolerance set for considering as linearly dependent atom [default tol=
            1.0000e-7]
   ind      (optional) indices determining the initial subspace
   []       can be used for  tol and ind

 Outputs:
   D        sub-dictionary extracted from D containing linearly independent
            atoms
   Di       indices of atoms in new D w.r.t to original D
   Q        orthonormal functions spanning the same space as new D
   beta     biorthogonal functions to new D

 References:
   L. Rebollo-Neira, "Dictionary redundancy elimination", IEE Proceedings - Vision, Image
   and Signal Processing, Vol(151,1), 31-34 (2004).

 See also DREOr DREOrp Biorthog.
```

5.2.7 DREOr

uses DRE method to produce and orthogonal basis from a dictionary and gives dictionary's indices of the atoms spannig the space.

```
 The difference with DREOrp is that it gives Di (see below)

 Implement column pivoting to choose a stable orthogonal basis from a given set, which
 could be redundant, called "dictionary"

 Usage:     [ Q, Di ] = DREOr( D, tol, ind );
            [ Q, Di ] = DREOr( D, tol );
            [ Q, Di ] = DREOr( D );

 Inputs:
```

```
  D       matrix the columns of which is a dictionary of normalized atoms
  tol     tolerance set for considering as linearly dependent columns [default tol=
          1.0000e-7]
  ind     (optional) indices determining the initial subspace
  []      can be used for tol and ind

Outputs:
  Q       orthonormal vectors spanning the same space as D (up to tol)
  Di      indices of linearly independent atoms that have been orthogonalized

References:
  L. Rebollo-Neira, "Dictionary redundancy elimination", IEE Proceedings - Vision, Image
  and Signal Processing, Vol(151,1), 31-34 (2004).

See also  DRE DREOrp and Biorthog
```

5.2.8 DREOrp

uses Dre method to produce and orthogonal basis from a dictionary.

```
The only difference with DREOr is that this only gives Q so it is a bit faster

implement column pivoting to choose an stable orthogonal basis from a given set, which
could be redundant, called "dictionary"

Usage:    [ Q ] = DREOr( D, tol, ind );
          [ Q ] = DREOr( D );

Inputs:
  D       matrix the columns of which is a dictionary of normalized atoms
  tol     tolerance set for considering as linearly dependent columns [default tol=
          1.0000e-7]
  ind     (optional) indices determining the initial subspace
  []      can be used for tol and ind

Outputs:
  Q       orthonormal vectors spanning the same space as D, up to tolerance tol

References:
  L. Rebollo-Neira, "Dictionary redundancy elimination", IEE Proceedings - Vision, Image
  and Signal Processing, Vol(151,1), 31-34 (2004).

See also DREOr DRE
```

5.2.9 FrDelete

deletes the requested vector from a given frame

```
It appropriately modifies the corresponding  duals "beta", the updating
depends on whether the vector "num" belongs to the remaining frame or not

Usage: [D,beta]=nfrdelete(D,beta,num);
Usage: [D,beta]=nfrdelete(D,beta,num,ic);

Inputs:
  D    frame
  beta dual frame
  num  index of  the vector to be eliminated
```

```
  ic = 1 if linearly independent  (linearly dependent otherwise)

Outputs:
  D    reduced  frame
  beta updated  dual frame (for the reduced D)

References:
 L. Rebollo-Neira, ‘‘Constructive updating/downdating of oblique projectors: a generalization of the G
Vol(40), 6381-6394 (2007).

See http://www.ncrg.aston.ac.uk/Projects/HNLApprox/ for more details
```

5.2.10 FrInsert

adds a vector to a frame. It also appropriately modifies the corresponding dual frame.

```
Usage: [D,beta]=nfrinsert(D,beta,atom,tol);

Inputs:
  D    frame
  beta dual  frame to D
  atom new  vector to be incorporated into the frame
  tol  tolerance (optional parameter) to decide linear dependence
       default value = 1.0000e-7
  dual if atom is linearly dependent dual is an arbitrary vector
        default dual=atom

Outputs:
  D    updated frame
  beta updated dual frame of D

References:
 [1] L. Rebollo-Neira, ‘‘Constructive updating/downdating of oblique projectors: a generalization of t
Journal of Physics A: Mathematical and Theoretical}, Vol(40), 6381-6394 (2007)

This routine is  equivalent to NBioInsert  if the new atom is independent

see http://www.ncrg.aston.ac.uk/Projects/HNLApprox/
```

5.2.11 FrInsertBlock

adds vectors to a basis and gives the duals spanning the same space

```
Usage: [D,beta]=FrInsertBlock(D,beta,gb,tol);

Inputs:
  D    basis
  gb   new vectors to be incorporated into the basis
  tol  tolerance (optional parameter) to decide linear dependence
       default value = 1.0000e-7

Outputs:
  D    updated basis
  beta updated biorthgonal basis


or http://www.ncrg.aston.ac.uk/Projects/HNLApprox/
```

5.2.12 NBioDictIns

enlarges a basis selected from a given dictionary by one vector. It modifies the biorthogonal basis for the same space and gives the projection of the remainding atoms onto the orthogonal complementary space

```
Usage:    lambda = NBioDictIns( D, lambda, k );

Inputs:
  D       dictionary, D(:,1-k-1) is the selected basis
  lambda  lambda(:,1:k-1)=biorthogonal functions to D(:,1:k-1)
          lambda(:,k:end)=D(:,k:end) without component from D(:,1:k-1)
  k       k-th atom of D to incorporate it says that first k-1 atoms are already in
          basis

Outputs:
  lambda  lambda(:,1:k)=biorthogonal functions to D(:,1:k)
          lambda(:,k+1:end)=D(:,k+1:end) without component from D(:,1:k)

References:
  L. Rebollo-Neira, "Recursive bi-orthogonalisation approach and orthogonal projectors",
  math-ph/0209026 (2002).

Note: the difference between this routine and BioDictIns is that in this routine the
orthonormal basis spanning D(:,1:k) is not available

See also BioDictIns, BioInsert, NBioInsert.
```

5.2.13 NBioInsert

adds an atom to a basis and modifies the biorthogonal basis for the same space

```
Usage:    [ D, beta ] = NBioInsert( D, beta, atom, tol );

Inputs:
  D       already selected basis
  beta    biorthogonal functions to D
  atom    new atom to be incorporated into basis
  tol     tolerance (optional parameter) to decide linear dependence default value =
          1.0000e-7

Outputs:
  D       updated basis
  beta    updated biorthogonal functions to D

References:
  L. Rebollo-Neira, "Recursive bi-orthogonalisation approach and orthogonal projectors",
  math-ph/0209026 (2002).

See also BioInsert, NBioDictIns, BioDictIns.
```

Chapter 6

Examples

6.1 Example-Summary

exa_ObliProj	using ObliProj
exa_OptObliProj	using OptObliProj
exa_RegObliProj	using RegObliProj

6.2 Example-Description

6.2.1 exa_ObliProj

using ObliProj

```
Separates the components f_v in V from f=f_v+f_w; with f_w in WC.
```

6.2.2 exa_OptObliProj

using OptObliProj

```
Separates the component fv in V  from f=fv+fw; with fw in WC.
```

6.2.3 exa_RegObliProj

using RegObliProj

```
Separates the component fv in V  from f=fv+fw; with fw in WC.
```

Part III

Image Processing Tools

6.3 Function-Summary

CalcPSNR	returns the PSNR between the original image and its approximation
DCos	generates a matrix whos columns are discrete cosine vectors.
DetectLines	returns the index of vertical impulsive lines in an image
GenerateHats	Generates a hat dictionary
GenerateTrapezium	generates a vector representing a trapezium
ImageApproximation	Returns an approximation of an image generated by choosing atoms from dictionary using either thresholding or a greedy algorithm.
RemoveDependantAtoms	removes any dependant atoms from a given dictionary
TranslatePrototype	translates a vector to construct either a dictionary or a basis

6.4 Function-Description

6.4.1 CalcPSNR

returns the PSNR between the original image and its approximation

```
Calculates the Peak Signal to Noise Ratio (PSNR) between 2 matrices containing
pixel intensity values.

Usage:            psnr = CalcPSNR( mImage1, mImage2 );

Inputs:
  mImage1         matrix of pixel intensity values representing the original image
  mImage2         matrix of pixel intensity values representing the approximated image
  maxIntensity    maximum allowed pixel intensity, defualt is 256 (8 bit image)

Outputs:
   psnr       the PSNR resulting from the  approximation
```

6.4.2 DCos

generates a matrix whos columns are discrete cosine vectors.

```
Returns discrete cosine vectors that belong to the Euclidean space of size szSpace.  The
deafult is to return a basis for the space.

Usage             mCosines = DCos( szSpace, nFrequencies, redundancy );
                  mCosines = DCos( szSpace, nFrequencies );
                  mCosines = DCos( szSpace );

Inputs:
  szSpace         the size of the Euclidean space the vectors should belong to
  nFrequencies    number of frequencies to use starting from 0.  If not specified will
                  be the same as the size of the space
  redundancy      redundancy of the dictionary, the default is 1 (basis)

Outputs:
  mCosines        matrix whos columns are discrete cosine vectors.
```

6.4.3 DetectLines

returns the index of vertical impulsive lines in an image

```
Searches a matrix representing the pixel intensities of an image for columns where all
the values are equal to lineValue.  Note this will also return the index of vertical
```

```
edges.

Usage:       iLine = DetectLines( mImage, lineValue );
             iLine = DetectLines( mImage );

Inputs:
  mImage     double matrix representing image pixel intensities
  lineValue  value to search for default is 0.

Outputs:
  iLine      vector containing the index's of the columns of mImage containing vertical
             lines
```

6.4.4 GenerateHats

Generates a hat dictionary

```
Builds a dictionary for a space of size szSpace from one or more hat dictionaries or
basis, where the length of there support is given in the vectot hats.

Usage:          mHatDictionary = GenerateHats( hats, szSpace, dictionary );

Inputs:
  hats            vector of support lengths for each dictionary
  szSpace         number of discrete points in each atom
  dictionary      set to 0 if the dictionary is composed from several hat basis, or 1 if
                  it is composed from several hat dictionaries

Outputs:
  mHatDictionary  matrix whose columns are the atoms fromt he hat dictionaries
```

6.4.5 GenerateTrapezium

generates a vector representing a trapezium

```
Generates a discrete vector of points representing the vertical distance between
the base of an isosceles trapezium and the other three sides. You choose the length
of the trapeziums base, this will 2 less than the size of vTrapezium as the base
values are zero. You also choose the length of the trapeziums top.

Usage:       vTrapezium = GenerateTrapezium( lBase, lTop );
             vTrapezium = GenerateTrapezium( lBase );

Inputs:
  lBase      number of discrete points for the trapeziums base
  lTop       number of discrete points for the trapeziums top, the default is 1

Outputs:
  vTrapezium column vector of points representing the vertical distance between
             the base of an isosceles trapezium and the other three sides.

See also TranslatePrototype
```

6.4.6 ImageApproximation

Returns an approximation of an image generated by choosing atoms from dictionary using either thresholding or a greedy algorithm.

```
Usage:           [ mImageApprox, mNCn, mCn, mICn, actualPSNR, processingTime, ...
                     mError ] = ImageApproximation( mImage, mAtoms, algorithm, ...
                     criteria, blockWidth );

Inputs:
  mImage         the matrix of pixel intensity values
  mAtoms         matrix whos columns are the atoms from the dictionary we use to
                 approximate the image with
  algorithm      name of the algorithm to use
  criteria       the target psnr between the original image and its approximation if using
                 a greedy algorithm or the threshold if using Thresholding
  blockWidth     width of the square blocks that the image will be processed in

Outputs:
  mImageApprox   matrix representing the approximated image
  mNCn           matrix containing the number of coefficients used to represent each
                 block of the image
  mICn           array containing the index's of the retained coefficients
  actualPSNR     psnr between the original image and its approximation
  processingTime time in seconds to process the image
  mError         matrix containing the norm of the error between the original image and
                 the approximated image for each block processed.
```

6.4.7 RemoveDependantAtoms

removes any dependant atoms from a given dictionary

```
Normalises the dictioanry and then removes any atoms that have an inner product of
within tol of 1.

Usage:               [ mUniqueDictionary iRemovedAtoms ] = RemoveDependantAtoms( ...
                         mDictionary, tol );
                     [ mUniqueDictionary iRemovedAtoms ] = RemoveDependantAtoms( ...
                         mDictionary);

Inputs:
  mDictionary        dictionary of atoms
  tol                tolerance of how similar atoms can be, default is 1e-13

Outputs:
  mUniqueDictionary  dictionary with dependant atoms removed
  iRemovedAtoms      index of the dependant atoms
```

6.4.8 TranslatePrototype

translates a vector to construct either a dictionary or a basis

```
Constructs a matrix whos columns are vectors forming either a redundant dictionary
or a basis for the Euclidean space of dimension szSpace. Each vector is generated
by translating one point at a time the discrete values contained in vPrototype,
i.e.
szSpace = 3;
vPrototype = [ 1 ];
mVectors = [ 1 0 0;
             0 1 0;
             0 0 1 ];
```

```
If dictionary is not specified or set to 1 we apply the 'cut off' approach to
create a dictionary for the space i.e.
dictionary = 1;
szSpace = 3;
vPrototype = [ 1 1];
mVectors = [ 1 0 0;
             1 1 0;
             0 1 1;
             0 0 1 ]
If dictionary is set to 0 we adopt cyclic boundry conditions to create a basis for
the space, i.e.
dictionary = 0;
szSPace = 3;
vPrototype = [ 1 1];
mVectors = [ 1 1 0;
             0 1 1;
             1 0 1 ];

Usage:        mVectors = TranslatePrototype( vPrototype, szSpace, dictionary );
              mVectors = TranslatePrototype( vPrototype, szSpace);

Inputs:
  vPrototype  vector representing the shape to be translated.
  szSpace     size of the Euclidean space we want to span.
  dictionary  0 to generate a basis and 1 to generate a redundant dictionary for
              the space, the default is to generate a dictionary.

Outputs:
  mVectors    matrix whos columns span the space of dimension szSpace

See also GenerateTrapezium
```

6.5 Examples

6.6 Example-Summary

exa_image_approximation	approximating an image using OMP
exa_impulse_removal	Elimination of random lines from an image using the function OOMP-FinalRefi()

6.7 Example-Description

6.7.1 exa_image_approximation

approximating an image using OMP

```
Example of using the OMP algorithm to approximate the image of Lena using a dictionary
comprised from a deiscrete cosine redundancy 2 dictionary and hat dictionaries of
support 1, 3 and 5.
```

6.7.2 exa_impulse_removal

Elimination of random lines from an image using the function OOMPFinalRefi()

```
Removes random vertical lines from an image by projecting onto atoms chosen from a
```

```
spline wavelets dictionary using OOMPFinalRefi().
Construct the Spline Wavelet Dictionary
```

Part IV

Spline Dictionaries

Chapter 7

Uniform

7.1 Function-Summary

BSpline	gives the analytical form of k-th B-spline of order m on l-th subinterval of given partition t
DictSpline	generates dictionary of cardinal B-spline functions of order m
Differ	calculates (s-1)-th divided difference of $\max((t-x)^(m-1), 0)$ where s is the number of knots in sequence t (i.e., s=length(t))
ErrorTest	tests orthogonality of a sequence or biorthogonality of two sequences.
Green	calculates $(x_+)^m = x^m$ for $x >= 0$ (it is 0 for $x < 0$). This functions is known as truncated powers.
NormDict	normalizes a given dictionary
SplineLevel	generates a B-spline dictionary depending on given parameters
SymSpline	gives the analytical form of B-spline of order m with knots in partition t
TSpline	generates B-spline basis of order m corresponding to knot sequence t

7.2 Function-Description

7.2.1 BSpline

gives the analytical form of k-th B-spline of order m on l-th subinterval of given partition t

```
Usage:   f = BSpline( m, k, l, t );
         f = BSpline( m, k, l );

Inputs:
  m      order of spline (m>=1), m=1 is a piecewise constant function
  k      specifies the index of spline, B-spline living on on I=[t(k),t(k+m)] is
         calculated if the sequence of knots t is not defined then
         I=t(k:k+m)=[k-1,k-1+m]
  l      specifies the interval [t(l),t(l+1)] on which the k-th B-spline is studied
  t      sequence of knots (optional)
         if t is a single number then the sequence is considered to be t:t+m
         if t not specified then t=k-1:k-1+m

Output:
  f      analytical form of k-th B-spline of order m on l-th subinterval of t

References:
  L.L. Schumaker, Spline Functions: Basic Theory, New York, Wiley, 1981.
```

```
Remark:   It is not recommended to use this procedure by user. You can do
          the same using SymSpline.
```

7.2.2 DictSpline

generates dictionary of cardinal B-spline functions of order m

```
Usage:    D = DictSpline( m, L, sp, b1, b2, type );

Inputs:
  m       order of spline functions (m>=1), m=1 is a piecewise constant function
  L       number of discrete points
  sp      vector [sp(1),sp(2)] specifying the interval
  b1      coarser partition of sp, points sp(1)+k*b1, k=0
  b2      finer partition of sp, points sp(1)+k*b2, k=0
  type    either ESEP or EPKB

Output:
  D       dictionary of cardinal B-spline functions of order

Remark:   This procedure uses the translation property of inner cardinal B-splines.
          It calls the routine SplineLevel to do the job.
```

7.2.3 Differ

calculates (s-1)-th divided difference of $\max((t-x)^(m-1), 0)$ where s is the number of knots in sequence t (i.e., s=length(t))

```
Usage:    f = Differ( x, t, m );

Inputs:
  x       dicrete variable
  t       knot sequence
  m       order

Output:
  f       (s-1)-th divided difference of max((t-x)^(m-1),0)

Remark:   Sequence t must be ordered.
```

7.2.4 ErrorTest

tests orthogonality of a sequence or biorthogonality of two sequences.

```
Usage:    errortest( D, beta );
          errortest( Q );

Inputs:
  D       sequence of vectors
  beta    (optional) biorthogonal sequence

Output:
  f       orthogonality of D or biorthogonality D w.r.t beta
```

7.2.5 Green

calculates $(x_+)^m = x^m$ for $x >= 0$ (it is 0 for $x < 0$). This functions is known as truncated powers.

```
Usage:    f = Green( x, m );

Inputs:
  x       discrete variable
  m       order

Output:
  f       (x_+)^m (see definition of this above)

References:
  L.L. Schumaker, Spline Functions: Basic Theory, New York-Wiley, 1981.
```

7.2.6 NormDict

normalizes a given dictionary

```
Usage:    D = NormDict( D, delta );
          D = NormDict( D );

Inputs:
  D       non-normalized dictionary
  delta   parameter, the discrete norm of D is multiplied by sqrt(delta)
          (default value is 1)
Outputs:
  D       normalized dictionary

Remark: It normalizes the columns of matrix D.
```

7.2.7 SplineLevel

generates a B-spline dictionary depending on given parameters

```
Usage:    D = SplineLevel( m, x, b1, b2, type );

Inputs:
  m       order of splines
  x       discrete variable
  b1      coarser partition of sp, points sp(1)+k*b1, k=0
  b2      finer partition of sp, points sp(1)+k*b2, k=0
  type    either ESEP or EPKB

Output:
  D       dictionary of spline functions

References:
   M. Andrle and L. Rebollo-Neira, "Cardinal B-spline dictionaries on a compact
   interval", Applied and Computational Harmonic Analysis, Vol(18,3), 336-346 (2005).
```

7.2.8 SymSpline

gives the analytical form of B-spline of order m with knots in partition t

```
Usage:    [ f, p ] = SymSpline( m, t );
          [ f, p ] = SymSpline( m );

Inputs:
  m       order of spline (m>=1), m=1 is a piecewise constant function
  t       sequence of knots (optional)
```

```
          if t is a single number then the sequence is considered to be t:t+m
          if t not specified, t=0:m

Outputs:
  f       array of m symbolical polynomials, each polynomial describes the desired
          B-spline on the subinterval [t(i),t(i+1)], i=1,...,m
  p       matrix of coefficients for polynomials expressed in f,
          each row in this matrix represents a_{m-1}, ..., a_0 for
          polynomials written as a_{m-1}x^{m-1}+...+ a_0

Remark:   If length of t is bigger than m+1, this procedure takes in account
          only first m+1 knots of t

References:
  L.L. Schumaker, Spline Functions: Basic Theory, New York, Wiley, 1981.
```

7.2.9 TSpline

generates B-spline basis of order m corresponding to knot sequence t

```
Usage:    D = TSpline( x, t, m );

Inputs:
  x       variable range (discrete points)
  t       knot sequence (could be multiple knots)
  m       order of splines (m=1 means a piecewise constant functions)

Output:
  D       B-spline function basis corresponding to knot sequence t

Remark:   The sequence t is sorted before calculation.

References:
  L.L. Schumaker, Spline Functions: Basic Theory, New York, Wiley, 1981.
```

Chapter 8

Non Uniform

8.1 Function-Summary

CutDic	produces a non-uniform B spline dictionary
NonBSpline	Computes the value of B spline basis at x0 by the recurrence formula for the non-uniform B spline.
NonUniB	computes all non-unfirom B spline over the partition 'p'
ProducePartition	using the curvature of the signal f, it compute the knots of a partition

8.2 Function-Description

8.2.1 CutDic

produces a non-uniform B spline dictionary

```
Usage:       D = CutDic( partition, m, L, level );

Inputs:
  partition   a partition to produce a non-uniform B spline dictionary (it can be
              computed by the function 'producepartition')
  m           the order of the spline
  L           the number of the sampling of the functions
  level       it is related with the width of the B spline basis. When level=1, the
              function outputs the B spline basis.

Outputs:
  D           the non-uniform B spline dictionary (matrix)
```

8.2.2 NonBSpline

Computes the value of B spline basis at x0 by the recurrence formula for the non-uniform B spline.

```
Usage:    re = NonBSpline( m, t, x0 );

Inputs:
  m       the order of splines
  t       the partition (the number of entris in t is m+1)
  x0      the point for computing

Outputs:
  re      the value of B spline at x0
```

8.2.3 NonUniB

computes all non-unfirom B spline over the partition 'p'

```
Usage:    D = NonUniB( a, b, m, p, L );

Inputs:
  a,b     the end points of the interval
  m       the order of the spline
  p       the partition
  L       the number of the sampling of the functions

Outputs:
  D       the non-uniform B spline over the partition p (matrix, D(:,j) denots jth
          B spline basis)
```

8.2.4 ProducePartition

using the curvature of the signal f, it compute the knots of a partition

```
Usage:        partition = ProducePartition( a, b, f, cut );

Inputs:
  a,b         the end points of the interval
  f           signal
  cut         sub-divided level of the partition.

Outputs:
  partition   the knots of the partition. The first entry is a and the last one is b
```

Chapter 9

Wavelets

9.1 Function-Summary

BuildDict	helps you to construct a multiresolution-like spline wavelet dictionary or a B-spline dictionary
ElimBound	eliminates redundant boundary wavelets to have a basis for the cut-off spline wavelet dictionary constructed with b=1
GDictFast	generates dictionaries by translating prototype functions
NumFun	states how many functions $f(a^j * x - b * k)$, where k is an integer, have non-trivial intersection with the interval $[c, d]$
SPL	generates B-splines of order m at scale j with translation parameter k, $B(a^j * x - k)$. Multiple knots for construction boundary functions are considered.
STPoint	returns such a translation parameter k that $f(a^j * x - b * k)$ is the first function having non-trivial intersection with the point c
ScalLevel	generates a set of all translated B-splines, $B(a^j * x - b * k)$, having non-trivial intersection with the given interval. It can be also used for construction (Chui) cubic B-spline basis (for $b = 1$ only).
SplineChuiWav	generates the Chui semi-orthogonal cubic spline wavelet, $w(a^j * x - k)$ on the given interval.
SplineScal	generates dilated/translated B-spline, $B(a^j * x - b * k)$, of order m on the given interval.
SplineWavelet	generates semi-orthogonal spline wavelet, $w(a^j * x - b * k)$, of order $1, 2$ or 4 on the given interval
WavLevel	generates a set of all translated spline wavelets, $w(a^j * x - b * k)$, having non-trivial intersection with the given interval. It can be also used for construction (Chui) cubic B-spline wavelet basis (for $b = 1$ only).

9.2 Function-Description

9.2.1 BuildDict

helps you to construct a multiresolution-like spline wavelet dictionary or a B-spline dictionary

```
To create a dictionary of your choice, edit this file and set the desired values of the
variables.

Available multiresolution-like dictionaries are:

name='1'           Haar dictionary (cut-off construction)
    ='2'           linear dictionary (cut-off construction)
```

```
    ='4'           cubic dictionary (cut-off construction)
    ='chui_cubic'  cubic spline wavelet basis (multiple knots)

Available B-spline dictionaries are:

name='1b'           piece-wise constant B-spline dictionary
    ='2b'           linear B-spline dictionary (cut-off construction)
    ='4b'           cubic B-spline dictionary (cut-off construction)
    ='Nb'           B-spline dictionary of order N (any positive integer)
    ='chui_cubicb'  cubic B-spline dictionary (multiple knots)


Description of other parameters

  a       dilation factor (positive integer)
  b       translation factor (b=1 for basis, keep 1/b being integer)
          use b=1 only for 'chui_cubic' and 'chui_cubicb'
  sp      array sp=[c,d] stands for interval [c,d]
  j       array containing all the scales to be considered
          note that larger 'j' implies finer scale
  L       number of points partitioning the interval sp=[sp(1) sp(2)]
          it must be a positive integer value satisfying
          L=J*(sp(2)-sp(1))/(b*(a^(-jmax)))+1 where jmax stands for the
          finest scale to be considered and J is a positive integer

Outputs:
  x       dicretization of the interval sp containing L nodes
  D       desired dictionary
  ind     array of indices separating different scales in D

EXAMPLE:
The following example constructs a cubic multiresolution-like wavelet dictionary D on
the interval [0,8] with scale factor a=2, b=0.5, j=[0:4] The interval [0,8] will be
partitioned into L=2049 points.

name='4';b=0.5;a=2;
sp=[0 8];
j=[0:4];
L=2049;
[D, ind] =gdictfast(name,{L;sp;j;a;b});

Comments:
For B-spline dictionaries see also TestSpline
For a cut-off multiresolution-like wavelet basis see ElimBound

References:
  M. Andrle and L. Rebollo-Neira, "Spline wavelet dictionaries for non-linear signal
  approximation", preprint, 2005.
  M. Andrle and L. Rebollo-Neira, "Cardinal B-spline dictionaries on a compact
  interval", Applied and Computational Harmonic Analysis, Vol(18,3), 336-346 (2005).
  C.K. Chui and E. Quak, "Wavelets on a Bounded Interval", in Numerical Methods of
  Approximation Theory, Vol. 9 (Eds. D. Braess and  L.L. Schumaker), pp. 53-75,
  Birkhauser, Basel, 1992.
```

9.2.2 ElimBound

eliminates redundant boundary wavelets to have a basis for the cut-off spline wavelet dictionary constructed with b=1

```
Usage:    [ DD, ind ] = ElimBound( m, D, ind );

Inputs:
  m       order of splines in dictionary (positive integer)
  D       dictionary
  ind     array of final indices for every scale in D (optional)

Outputs:
  DD      basis, D without the redundant boundary wavelets
  ind     updated indices for DD

For the construction of cut-off multiresolution-like wavelet dictionaries see BuildDIct.

Note: The first scale is assumed to contain scaling functions thus the redundant
wavelets are removed from all scales except the first one.

If ind is not specified it considers only one scale (wavelets) and m-1 functions are
removed from both sides.

Comments: The support of semi-orthogonal spline wavelet of order m is given by
supp=2*m-1. Thus the number of functions to eliminate at every scale at each border is
n=(supp-1)/2=m-1.

References:
  M. Andrle and L. Rebollo-Neira, "Spline wavelet dictionaries for non-linear signal
  approximation", preprint, 2005.
```

9.2.3 GDictFast

generates dictionaries by translating prototype functions

```
Usage:    D = GDictFast( name, pars );

Inputs:
  name    type of dictionary (string format)
  pars    parameters (cell format)
          pars = {L,sp,j,a,b}
Description of the parameters
  L       number of points partitioning the interval sp=[sp(1) sp(2)]
  sp      array sp=[c d] stands for interval [c,d]
  j       array containing all the scales to be considered note that larger j implies
          finer scale
  a       dilation factor (positive integer)
  b       translation factor (b=1 for basis, keep 1/b being integer) for 'chui_cubic'
          and 'chui_cubicb' use b=1 only

Outputs:
  D       desired dictionary
  ind     array of indices separating different scales in D

Comments:
The dictionaries are constructed in the following way: a prototype function for every
scale is computed for values x=linspace(sp(1),sp(2),L) (x is a dicretization of the
```

```
interval sp containing L nodes). Then this prototype function is shifted on x by an
appropriate number of nodes. For this end  (L-1)*(b*a^(-jmax))/(sp(2)-sp(1)) must be
an integer where jmax stands for the finest scale parameter. In the case of 'chui_cubic'
or 'chui_cubicb' dictionaries only the inner functions are constructed by the method
above. The boundary function are calculated using given analytical expressions.

Only scales with at least one inner function are considered.

References:
  M. Andrle and L. Rebollo-Neira, "Spline wavelet dictionaries for non-linear signal
  approximation", preprint, 2005.
  M. Andrle and L. Rebollo-Neira, "Cardinal B-spline dictionaries on a compact interval",
  Applied and Computational Harmonic Analysis, Vol(18,3) 336-346 (2005).
```

9.2.4 NumFun

states how many functions $f(a^j * x - b * k)$, where k is an integer, have non-trivial intersection with the interval $[c, d]$

```
Usage:    n = NumFun( name, type, c, d, j, a, b );

Inputs:
  name    order of splines (positive integer) or 'chui_cubic' (string)
  type    string, 'scal.f.' for scaling functions, 'wavelet' for wavelets
  c,d     the given interval [c,d]
  j       scale level (a^j is the dilation)
  a       scaling factor
  b       translation factor

Output:
  n       the number of consecutive translates f(a^j*x-b*k) having non-trivial intersection
          with the interval [c,d]
```

9.2.5 SPL

generates B-splines of order m at scale j with translation parameter k, $B(a^j * x - k)$. Multiple knots for construction boundary functions are considered.

```
Usage:    f = SPL( x, m, j, k, a );

Inputs:
  x       a dicretization of the given interval
  j       scale level (a^j is the dilation)
  m       order of splines
  k       translation parameter
  a       scale factor
Output:
  f       B-splines of order m at scale j with translation parameter k

Note:
The interval [x(1) x(end)] must be [0,K] type where K is an integer
```

9.2.6 STPoint

returns such a translation parameter k that $f(a^j * x - b * k)$ is the first function having non-trivial intersection with the point c

```
Usage:    k = STPoint( name, type, c, j, a, b );
```

```
Inputs:
  name    order of splines (positive integer) or 'chui_cubic' (string)
  type    string, 'scal.f.' for scaling functions, 'wavelet' for wavelets
  c       left point of the given interval
  j       scale level (a^j is the dilation)
  a       scaling factor
  b       translation factor

Output:
  k       desired translation parameter such that f(a^j*x-b*k) is the first function
          having non-trivial intersection with the point c
```

9.2.7 ScalLevel

generates a set of all translated B-splines, $B(a^j * x - b * k)$, having non-trivial intersection with the given interval. It can be also used for construction (Chui) cubic B-spline basis (for $b = 1$ only).

```
Usage:    D = ScalLevel( name, x, j, a, b )

Inputs:
  name    specifies the type of splines
  x       a dicretization of the given interval
  j       scale level (a^j is the dilation)
  a       scaling factor
  b       translation factor

Available B-splines are:
name=1             piece-wise constant B-spline dictionary
    =2             linear B-spline dictionary (cut-off construction)
    =4             cubic B-spline dictionary (cut-off construction)
    =N             B-spline dictionary of order N (any positive integer)
    ='chui_cubic'  cubic B-spline dictionary (multiple knots)

Output:
  D       set of B-splines at scale j having non-trivial intersection with the given
          interval
```

9.2.8 SplineChuiWav

generates the Chui semi-orthogonal cubic spline wavelet, $w(a^j * x - k)$ on the given interval.

```
Usage:    w = SplineChuiWav( x, j, k, a );

Inputs:
  x       a dicretization of the given interval
  j       scale level (a^j is the dilation)
  k       translation parameter
  a       scale factor
Output:
  w       normalized cubic Chui semi-orthogonal spline wavelet

References:
  C.K. Chui and E. Quak, "Wavelets on a Bounded Interval", in Numerical Methods of
  Approximation Theory, Vol. 9 (Eds. D. Braess and L.L. Schumaker), pp. 53-75,
  Birkhauser, Basel, 1992.
```

9.2.9 SplineScal

generates dilated/translated B-spline, $B(a^j * x - b * k)$, of order m on the given interval.

```
Usage:    w = SplineScal( x, j, k, m, a, b

Inputs:
  x       a dicretization of the given interval
  j,k     dilation and translation parameters
  a       scaling factor
  b       translation factor
  m       order of spline (positive integer)

Output:
  w       dilated/translated B-spline of a given order
```

9.2.10 SplineWavelet

generates semi-orthogonal spline wavelet, $w(a^j * x - b * k)$, of order $1, 2$ or 4 on the given interval

```
Usage:    w = SplineWavelet( x, j, k, m, a, b );

Inputs:
  x       a dicretization of the given interval
  j,k     dilation and translation parameters
  a       scaling factor
  b       translation factor
  m       order of spline wavelet (1,2 or 4)

Output:
  w       normalized dilated/translated spline wavelet of a given order
```

9.2.11 WavLevel

generates a set of all translated spline wavelets, $w(a^j * x - b * k)$, having non-trivial intersection with the given interval. It can be also used for construction (Chui) cubic B-spline wavelet basis (for $b = 1$ only).

```
Usage:    D = WavLevel( name, x, j, a, b );

Inputs:
  name    specifies the type of spline wavelets
  x       a dicretization of the given interval
  j       scale level (a^j is the dilation)
  a       scaling factor
  b       translation factor

Available types of spline wavelets:
name=1            Haar dictionary (cut-off construction)
    =2            linear dictionary (cut-off construction)
    =4            cubic dictionary (cut-off construction)
    ='chui_cubic' cubic spline wavelet basis (multiple knots)

Output:
  D       set of all normalized spline wavelets at scale level j having non-trivial
          intersection with the given interval
```