Construction of wavelet dictionaries for ECG modelling

Dana Černá

Department of Mathematics and Didactics of Mathematics, Technical University of Liberec, Studentská 2, 461 17 Liberec, Czech Republic

Laura Rebollo-Neira

Mathematics Department, Aston University, B4 7ET, Birmingham, UK

Abstract

Algorithms and MATLAB implementation facilitating the construction of wavelet dictionaries for ECG modeling are presented. The model enables the representation of an ECG record as a linear combination of fewer elementary components than those required by a wavelet basis. That result was demonstrated in a previous work on the MIT-BIH Arrhythmia data set. This paper complements the previous one. It produces the technical details of the methods and algorithms which are implemented by the software for constructing different families of wavelet dictionaries. The implementation allows for straightforward further extensions to include additional wavelet families.

Keywords: ECG modeling, wavelet dictionaries, dimensionality reduction
2010 MSC: 92C55, 65T60, 94A12

1. Introduction

The electrocardiogram (ECG) is a routine test for clinical medicine. It plays a crucial role in the diagnosis of a broad range of anomalies in the human heart; from arrythmias to myocardial infarction.

The widely available digital ECG data has facilitated the development of algorithms for ECG processing and interpretation. In particular, the literature for computerized arrhythmia detection and classification is extensive. Useful review matterial [21, 22] can help with the introduction to state of the art techniques, which nonetheless keeps growing [1, 5, 7, 20, 26].

A common first step in ECG modeling consists in reducing the dimensionality of the signal. This entails to represent the informational content of the record by means of significantly fewer parameters than the number of samples in the digital ECG. When the aim is to reproduce the original signal at low level distortion, the step is frequently realized through transformations such as the Wavelet Transform and the Discrete Cosine Transform.

In the last few years alternative approaches, falling within the category of sparse representation of ECG signals, have been considered. Within the sparse representation framework, an ECG record is represented as a linear combination of elementary components, called *atoms,* which are selected from a redundant set, called a *dictionary.* The success of the methods developed within this framework depends on both, the selection technique and the proposed dictionary. The selection techniques which are widely applied for sparse representation of general signals are either greedy pursuit strategies [24, 27, 30], or strategies based on minimization of the 1-norm as a cost function [10]. Suitable dictionaries depend on the class of signals being processed. These can be designed at hoc or be learned from training data. Sparse representation of ECG signals has been tackled by both these approaches, e.g. [2] learns dictionaries using some part of the records for ECG compression and [28] uses Gabor dictionaries for structuring features for classification.

In a recent publication [29] we have shown that wavelet dictionaries, derived from known wavelet families, are suitable for representing an ECG record as a linear combination of fewer elementary components than those required by a wavelet basis. The model was shown to be successful for dimensionality reduction and lossy compression. As far as compression is concerned the method advanced in [29] produces compression results improving upon previously reported benchmarks [19, 23, 25, 34] for the MIT-BIH Arrhythmia data set without pre-processing. With regard to dimensionality reduction, wavelet dictionaries considerably improve upon the results achieved with the wavelet basis of the same family [29, 32]. This result motivated the present Communication. While in [29] the dictionaries have been used to demonstrate their suitability for dimensionality reduction of ECG signals at low level distortion, the details of their numerical construction were not given. This paper complements the previous work by presenting the algorithms for building dictionaries from the following mother wavelet prototypes:

1) Chui-Wang linear spline wavelet [11]
2) Chui-Wang quadratic spline wavelet [11]
3) Chui-Wang cubic spline wavelet [11]
4) primal CDF97 wavelet [6]
5) dual CDF97 wavelet [6]
6) primal CDF53 wavelet [6]
7) linear spline wavelet with short support and 2 vanishing moments [9, 18]
8) quadratic spline wavelet with short support and 3 vanishing moments [9, 18]
9) cubic spline wavelet with short support and 4 vanishing moments [9, 18]
10) Daubechies wavelet with 3 vanishing moments [16]
11) Daubechies wavelet with 4 vanishing moments [16]
12) Daubechies wavelet with 5 vanishing moments [16]
13) symlet with 3 vanishing moments [17]
14) symlet with 4 vanishing moments [17]
15) symlet with 5 vanishing moments [17]
16) coiflet with 2 vanishing moments and support of length 6 which is the most regular [17]

17) coiflet with 3 vanishing moments and support of length 8 [17]

The method proposed in [29] for modeling a given ECG signal proceeds as follows. Assuming that the signal is given as an N-dimensional array, this array is partitioned into Q cells $\mathbf{f}^{\mathrm{c}}\{q\}, q = 1, \ldots, Q$. Thus, each cell $\mathbf{f}^{\mathrm{c}}\{q\}$ is an N_b-dimensional vector, which is modeled by an atomic decomposition of the form

$$\mathbf{f}^{\mathrm{a}}\{q\} = \sum_{n=1}^{k(q)} \mathbf{c}\{q\}(n)\mathbf{d}_{\ell\{q\}(n)}. \tag{1}$$

For each cell q, the atoms $\mathbf{d}_{\ell_{\{q\}(n)}}$, $n = 1, \ldots, k(q)$ are selected from a dictionary through the greedy Optimized Orthogonal Matching Pursuit (OOMP) algorithm [30, 31]. The array $\boldsymbol{\ell}\{q\}$ is a vector whose components $\ell\{q\}(n)$, $n = 1, \ldots, k(q)$ contain the indices of the selected atoms for decomposing the q-th cell in the signal partition. The OOMP method, for selecting these indices and computing the corresponding coefficients $\mathbf{c}\{q\}$, $n = 1, \ldots, k(q)$ in (1), is fully implemented by the OOMP function included as a tool of the software.

Each of the proposed dictionaries consists of two components. One of the components contains a few elements, say M_c, from a discrete cosine basis. This component of the dictionary allows for the fact that ECG signals are normally superimposed to a smooth background. It is given as a $N_b \times M_c$ matrix $\mathbf{D}^C$. The other component is the wavelet-based dictionary, which is given as a $N_b \times M_w$ matrix $\mathbf{D}^W$. Thus, the whole dictionary $\mathbf{D}$ is an $N_b \times (M_c + M_w)$ matrix obtained by the horizontal concatenation of $\mathbf{D}^C$ and $\mathbf{D}^W$. The next section is dedicated to the construction of $\mathbf{D}^W$.

The paper is organized as follows. Sec. 2 gives all the details for the construction of different wavelet prototypes and the concomitant wavelet dictionaries generated by those prototypes. Secs 3, 4 and 5 deliver details and examples of the use of the MATLAB software for modeling ECG signals within the proposed framework.

The software has been made available on a dedicated webpage [15]. The implementation allows for straightforward further extension of the options for wavelet types.

2. Method for construction of wavelet dictionaries

In this section we produce all the pseudo-codes for the construction of wavelets dictionaries, which can be used to achieve the model of every segment in a signal partition. As already mentioned, each dictionary is obtained by taking the prototypes from a wavelet basis and translating them within a shorter step than that corresponding to the wavelet basis.

Throughout the paper we adopt the following notation. Boldface fonts are used to indicate Euclidean vectors and matrices. Standard mathematical fonts are used to indicate components, e.g., $\mathbf{d} \in \mathbb{R}^N$ is a vector of N-components $d(i) \in \mathbb{R}$, $i = 1, \ldots, N$ and $\mathbf{D} \in \mathbb{R}^{N \times M}$ is a matrix of elements $D(i, j)$, $i =$

$1\ldots,N, j=1,\ldots,M$. The symbol $L^2(\mathbb{R})$ denotes the space of square integrable functions.

Wavelets are usually constructed starting from a *multiresolution analysis*, which is a sequence $\{V_j\}_{j=j_0}^{\infty}$ of closed subspaces of the space $L^2(\mathbb{R})$ which are nested and their union is dense in $L^2(\mathbb{R})$, i.e.,

$$V_j \subset V_{j+1} \quad \forall\, j \geq j_0, \quad \overline{\bigcup_{j=j_0}^{\infty} V_j} = L^2(\mathbb{R}). \tag{2}$$

We assume that there exists a function $\phi \in L^2(\mathbb{R})$ such that for $j \geq j_0$ functions

$$\phi_{j,k}(x) = 2^{j/2}\phi\left(2^j x - k\right), \quad k \in \mathbb{Z}, \tag{3}$$

form uniformly stable bases of the spaces V_j, i.e., the bases are Riesz bases with bounds independent of the level j, see e.g. [8]. The functions $\phi_{j,k}$ are called *scaling functions* and the function ϕ is called a *generator* of scaling functions. Next we present a method for the actual construction of the scaling functions.

2.1. Generation of scaling functions

We assume that ϕ has a compact support $[0,K]$ for some $K \in \mathbb{N}$. From the nestedness of the multiresolution spaces V_j, it follows that there exists a *scaling filter* $\mathbf{h} = (h(1),\ldots,h(K+1))$ such that

$$\phi(x) = \sum_{k=1}^{K+1} h(k)\,\phi(2x+1-k) \quad \forall\, x \in \mathbb{R}. \tag{4}$$

If $\int_0^K \phi(x)\,dx = c \neq 0$ then, integrating (4), we obtain

$$c = \sum_{k=1}^{K+1} h(k)\,\frac{c}{2} \tag{5}$$

which implies that $\mathbf{h}$ has to be normalized such that

$$\sum_{k=1}^{K+1} h(k) = 2. \tag{6}$$

The scaling equation (4) enables computing values of the scaling function ϕ at points $k/2^u$ for $k=0,\ldots,K2^u$, $u \in \mathbb{N}$. First we compute values of ϕ at integer points. Since $\operatorname{supp}\phi = [0,K]$, we have $\phi(k) = 0$ for $k \neq (0,K)$. Let us define a vector

$$\Phi = (\phi(0),\ldots,\phi(K-1))^T, \tag{7}$$

where the $(.)^T$ indicates the transpose operation. Substituting $x = 0,\ldots,K-1$, into (4), we obtain

$$\begin{aligned}\Phi(i) &= \phi(i-1) = \sum_{k=1}^{K+1} h(k)\,\phi(2i-1-k) \qquad (8)\\ &= \sum_{j=2i-1}^{2i-K-1} h(2i-j)\,\phi(j-1) = \sum_{j=2i-1}^{2i-K-1} h(2i-j)\,\Phi(j)\,.\end{aligned}$$

We set $h(k) = 0$ for $k < 1$ and $k > K+1$ and define a matrix $\mathbf{A}$ by

$$A(i,j) = h(2i-j)\,, \quad i,j = 1,\ldots,K. \qquad (9)$$

Then, (8) is equivalent to

$$\Phi = \mathbf{A}\Phi. \qquad (10)$$

This means that Φ is an eigenvector corresponding to the eigenvalue 1 of the matrix $\mathbf{A}$. If the multiplicity of this eigenvalue is 1, then Φ is given uniquely up to a multiplication by a constant. Our aim is to compute a vector **phi** such that

$$\mathrm{phi}(m) = \phi\left(\frac{m-1}{2^u}\right), \quad m = 1,\ldots,K\,2^u+1, \qquad (11)$$

for a chosen level $u \in \mathbb{N}$. From (7) and (11) we have

$$\mathrm{phi}(2^u(l-1)+1) = \phi(l-1) = \Phi(l)\,, \quad l = 1,\ldots,K. \qquad (12)$$

We compute values of ϕ at points $l/2$. Note that for l even we already know these values. Using (4) and (12) we obtain

$$\begin{aligned}\mathrm{phi}\left(l2^{u-1}+1\right) &= \phi\left(\frac{l}{2}\right) = \sum_{k=1}^{K+1} h(k)\,\phi(l+1-k) \qquad (13)\\ &= \sum_{k=1}^{K+1} h(k)\,\mathrm{phi}\left((l+1-k)\,2^u+1\right)\end{aligned}$$

for $l = 1,3,\ldots,2N-1$. Similarly, we compute values of ϕ at points $l/4$, and thus we continue until we determine values at points $l/2^u$. More precisely, for $i = 1,\ldots,u$ we assume that we know values of ϕ at $l/2^{i-1}$, $l = 0,\ldots,K2^{i-1}+1$, and we compute the values

$$\mathrm{phi}(m) = \phi(x)\,, \quad m = x2^u+1, \quad x = l/2^{i-1}+1/2^i. \qquad (14)$$

Using (4) we obtain

$$\begin{aligned} \mathrm{phi}\,(m) &= \phi\,(x) = \sum_{k=1}^{K+1} h\,(k)\,\phi\,(2x+1-k) \\ &= \sum_{k=1}^{K+1} h\,(k)\,\mathrm{phi}\,((2x+1-k)\,2^u+1)\,. \end{aligned} \tag{15}$$

Remark 1. Some scaling functions such as spline scaling functions are known in an explicit form and their values can be evaluated directly. However, an advantage of our approach is that it is more general and can be used for a large class of wavelet families.

2.2. Construction of wavelet generators from scaling functions

Let W_j be complement spaces such that $V_{j+1} = V_j \oplus W_j$, where $\oplus$ denotes a direct sum. Wavelet functions $\psi_{j,k}$ are constructed in the form:

$$\psi_{j,k}\,(x) = 2^{j/2}\psi\left(2^j x - k\right), \quad k \in \mathbb{Z}, \tag{16}$$

to be a basis for W_j and such that

$$\mathcal{B} = \{\phi_{j_0,k}, k \in \mathbb{Z}\} \cup \{\psi_{j,k}, k \in \mathbb{Z}, j \geq j_0\} \tag{17}$$

called a *wavelet basis*, is a Riesz basis of the space $L^2\,(\mathbb{R})$.

Since $W_j \subset V_{j+1}$ there exists a vector $\mathbf{g} = (g(1), \ldots, g(M+1))$ such that

$$\psi\,(x) = \sum_{k=1}^{M+1} g\,(k)\,\phi\,(2x+1-k)\,. \tag{18}$$

The vector $\mathbf{g}$ is called a *wavelet filter*. From (18) we have

$$\mathrm{supp}\,\psi = \left[0, \frac{M+K}{2}\right]. \tag{19}$$

In Algorithm 1 we compute a vector **psi** such that

$$\mathrm{psi}\,(m) = \psi\left(\frac{m-1}{2^u}\right), \quad m = 1, \ldots, (M+K)\,2^{u-1}+1, \tag{20}$$

in the following way. Due to (18) and (20), we have

$$\begin{aligned} \mathrm{psi}\,(m) &= \psi\left(\frac{m-1}{2^u}\right) = \sum_{k=1}^{M+1} g\,(k)\,\phi\left((m-1)\,2^{1-u}+1-k\right) \\ &= \sum_{k=1}^{M+1} g\,(k)\,\mathrm{phi}\,(2m-1+(1-k)\,2^u)\,. \end{aligned} \tag{21}$$

The sum in the last equation is computed as a cyclic sum. For

$$m = 1, \ldots, (M + K)2^{u-1} + 1 \tag{22}$$

we set $\mathrm{psi}\,(m) = 0$ and for $k = 1, \ldots, M + 1$ we do

$$\mathrm{psi}\,(m) = \mathrm{psi}\,(m) + g\,(k)\,\mathrm{phi}\,(2m - 1 + (1 - k)\ 2^u)\,, \tag{23}$$

if $1 \leq 2m - 1 + (1 - k)\ 2^u \leq K2^u + 1$. Using the substitution

$$2m - 1 + (1 - k)\ 2^u\ = 2i - 1, \tag{24}$$

for $i = 1, \ldots, K2^{u-1} + 1$, we obtain

$$\mathrm{psi}\left(i + (k - 1)\,2^{u-1}\right) = \mathrm{psi}\left(i + (k - 1)\,2^{u-1}\right) + g\,(k)\,\mathrm{phi}\,(2i - 1)\,. \tag{25}$$

Algorithm 1 computes vectors **phi** and **psi** for given scaling and wavelet filters. The filters corresponding to the wavelet families supported by the software are given in Appendix A (Algorithm 8).

Algorithm 1
Procedure [**phi**,**psi**] = WaveletGen(**h**,**g**,u)

Input:

h	scaling filter
g	wavelet filter
u	level (integer) that determines points $l/2^u$

Output:

phi	vector of scaling function values (c.f. (11))
psi	vector of wavelet values (c.f. (20))

$K = \mathrm{length}(\mathbf{h}) - 1$ {support length of ϕ}
$\mathbf{h} = 2\,\mathbf{h}/\mathrm{sum}\,(\mathbf{h})$ {normalization of **h** (c.f. (6))}
{Compute a matrix **A** using (9)}
$\mathbf{A} = \mathrm{zeros}(K)$
for $i = 1 : K$ **do**
  for $j = 1 : K$ **do**
    if $1 \leq 2i - j \leq K + 1$ **then**
      $A\,(i, j) = h\,(2i - j)$
    end if
  end for
end for
{Compute eigenvalues and eigenvectors of the matrix **A**}
[V,D]=eig(**A**)
{Find an index of a column corresponding to eigenvalue 1}
$k = 0$ {k is the multiplicity of eigenvalue 1}
for $i = 1 : K$ **do**
  if $|(D(i, i) - 1| < 10^{-7}$ **then**

```
      column=i; k = k + 1
    end if
  end for
  if  k ≠ 1  then
    error('Impossible to construct scaling function: eigenvalue 1 must have
    multiplicity 1')
  else
    phi = zeros(K2^u + 1, 1)
    {Eigenvector V(:, column) represents values of φ at integer points}
    phi(1 : 2^u : (K − 1)2^u + 1) = V(:, column)  {c.f. (12)}
    {Compute values of φ at points l/2^u}
    for  i = 1 : u  do
      for  l = 1 : K2^(i−1)  do
        x = 2^(−i) + (l − 1) 2^(−i+1); m = x 2^u + 1 {c.f. (14)}
        for  k = 1 : K + 1  do
          if  0 ≤ 2x − k + 1 ≤ K then
            phi(m) = phi(m) + h(k)phi((2x − k + 1) 2^u + 1)  {c.f. (15)}
          end if
        end for
      end for
    end for
    M = length(g) − 1
    {Compute psi containing values of ψ at points l/2^u}
    psi = zeros((K + M) 2^(u−1) + 1, 1)
    for  k = 1 : M + 1  do
      i1 = (k − 1) 2^(u−1) + 1; i2 = (k − 1) 2^(u−1) + 1 + K 2^(u−1)
      psi(i1 : i2) = psi(i1 : i2) + g(k) phi(1 : 2 : K 2^u + 1)  {c.f. (25)}
    end for
  end if
```

2.3. Construction of wavelet bases and dictionaries

Hereafter we drop all normalization factors and normalize all the vectors once they have been constructed. Note that in (17) we used a translation parameter $k \in \mathbb{Z}$ and since $\mathcal{B}$ is a Riesz basis the functions from $\mathcal{B}$ are linearly independent. Now, we choose a parameter b such that $b = 2^{-m}$ for some integer m. We define functions

$$\phi_{j_0,k,b}\left(x\right) = \phi\left(2^{j_0}x - bk\right), \quad k \in \mathbb{Z}, \tag{26}$$

and

$$\psi_{j,k,b}\left(x\right) = \psi\left(2^{j}x - bk\right), \quad k \in \mathbb{Z}, \quad j \geq j_0, \tag{27}$$

which form a redundant dictionary [3, 4, 33]. Obviously, $b = 1$ corresponds to a basis.

The left graph of Figure 1 shows two consecutive wavelet functions taken from a linear spline bases [11]. The right graph of Figure 1 corresponds to two consecutive wavelet functions taken from the dictionary spanning the same space which corresponds to $b = 1/4$.

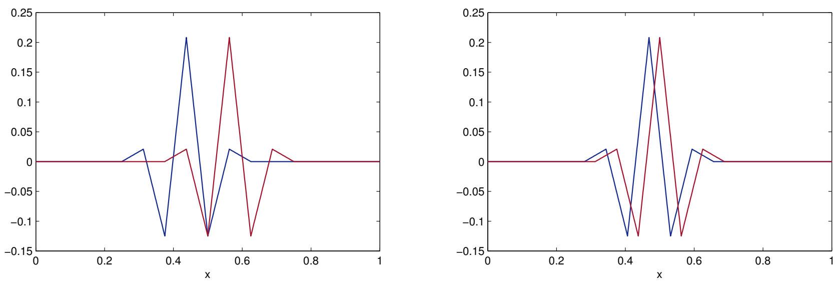

Figure 1: Wavelet functions taken from a basis (left) and a dictionary (right) corresponding to a linear spline-wavelet prototype from [11].

Algorithm 2 constructs a discrete dictionary, i.e., a matrix $\mathbf{D}^W$ which contains values of functions from (26) and (27) at N_b equidistant points for some chosen levels determined by the vector $\mathbf{j}$. Since Algorithm 1 enables us to construct values at points of the form $l/2^u$, we evaluate functions (26) and (27) at the points

$$x = \frac{l}{2^r}, \quad l = 0, \ldots, N_b - 1, \quad r = \left\lceil \frac{\log(N_b - 1)}{\log(2)} \right\rceil, \tag{28}$$

where $\lceil y \rceil$ denotes the smallest integer number larger than y.

For a chosen vector of levels $\mathbf{j}$, we define a vector of indices **ind** such that ind(1) is the number of scaling functions at level $j(1)$, and ind(l) is the number of wavelets at level $j(l-1)$ for $l = 1, \ldots, J$, where J is the length of $\mathbf{j}$. We have

$$\operatorname{supp} \phi_{j(1),k,b} = \left[\frac{bk}{2^{j(1)}}, \frac{bk + K}{2^{j(1)}} \right], \quad \operatorname{supp} \psi_{j,k,b} = \left[\frac{bk}{2^j}, \frac{bk + \frac{K+M}{2}}{2^j} \right]. \tag{29}$$

Comparing the supports of these functions and the interval

$$I = \left[0, \frac{N_b - 1}{2^r} \right] \tag{30}$$

which contains the points from (28), we find that the number of inner scaling functions, i.e., scaling functions with the whole support in I, is

$$n_i = \left\lfloor \frac{(N_b - 1)\, 2^{j(1)-r} - K}{b} \right\rfloor + 1, \tag{31}$$

where the symbol $\lfloor y \rfloor$ denotes the largest integer number smaller than y. The number of left boundary scaling functions, i.e., functions that have only a part of the support in the interior of I and their support contains 0, is

$$n_1 = Ka - 1, \quad a = 1/b, \tag{32}$$

and similarly the number of right boundary scaling functions is

$$n_2 = \left\lceil \frac{(N_b - 1)\ 2^{j(1)-r}}{b} \right\rceil - \left\lfloor \frac{(N_b - 1)\ 2^{j(1)-r} - K}{b} \right\rfloor - 1. \tag{33}$$

Hence, we have

$$\operatorname{ind}(1) = n_1 + n_i + n_2 = Ka - 1 - \left\lceil \frac{(N_b - 1)\ 2^{j(1)-r}}{b} \right\rceil. \tag{34}$$

Similarly, the number of wavelet functions on the level $j(l)$ is

$$\operatorname{ind}(1 + l) = sa - 1 + \left\lceil \frac{(N_b - 1)\ 2^{j(l)-r}}{b} \right\rceil. \tag{35}$$

The first ind(1) columns of $\mathbf{D}^W$ contain values of scaling functions (26), which restricted to I are not identically zero, at points given in (28), i.e.,

$$D^W(k, l) = \phi\left(2^{j(1)} \frac{k - 1}{2^r} - b\,(l - Ka)\right) \tag{36}$$

for $k = 1, \ldots, N_b$, $l = 1, \ldots, \operatorname{ind}(1)$. The above equation can be recast:

$$\begin{aligned} D^W(k, l) &= \phi\left(\frac{(k-1) - b\,(l - Ka)\,2^{r-j(1)}}{2^{r-j(1)}}\right) \\ &= \operatorname{phi}\left(k - b\,(l - Ka)\,2^{r-j(1)}\right), \end{aligned} \tag{37}$$

where **phi** is defined by (11) for the level $u = r - j(1)$. Using the substitution $m = k - b\,(l - Ka)\,2^{r-j(1)}$, we obtain

$$D^W\left(m + b\,(l - Ka)\,2^{r-j(1)}, l\right) = \operatorname{phi}(m), \quad m = 1, \ldots, K2^{r-j(1)} + 1, \tag{38}$$

under the assumption that $1 \leq m + b\,(l - Ka)\,2^{r-j(1)} \leq N_b$.

The other columns of $\mathbf{D}^W$ contain values of wavelet functions (27) for levels $j = j(1), \ldots, j(J)$ at points (28), i.e.,

$$D^W(k, n_p + l) = \psi\left(2^j \frac{k - 1}{2^r} - b\,(l - sa)\right), \quad s = \frac{M + K}{2}, \tag{39}$$

for $k = 1, \ldots, N_b$, $l = 1, \ldots, \operatorname{ind}(j + 1)$, and $n_p = \sum_{p=j(1)}^{j} \operatorname{ind}(p + 1 - j(1))$.

Similarly as above we obtain

$$D^W \left(m + b\,(l - sa)\ 2^{r-j(l)}, l + n_p \right) = \mathrm{psi}(m), \tag{40}$$

where $1 \le m + b\,(l - sa)\,2^{r-j(1)} \le N_b$ and **psi** is defined by (20) for the level $u = r - j(1)$.

The following procedure WaveletDict computes a wavelet dictionary.

Algorithm 2
Procedure $[\mathbf{D}^W, \mathbf{ind}, \mathbf{col}]$ = WaveletDict(namef, N_b, $\mathbf{j}$, b)

Input:

namef	name of a wavelet family, for available choices see Appendix A
N_b	number of points
$\mathbf{j}$	vector of levels
b	translation factor $b = 2^{-r_b}$ for some integer r_b

Output:

$\mathbf{D}^W$	wavelet dictionary
ind	ind(1) is the number of scaling functions at level $j(1)$, and ind(k) for $k > 1$ is the number of wavelets at level $j(k-1)$
col	cell array such that col$\{n\} = \{j, k, \mathrm{type}, \mathrm{function}\}$ if the n-th column of $\mathbf{D}^W$ corresponds to values of a scaling function $\phi(2^j x - bk)$ or a wavelet $\psi(2^j x - bk)$; type='inner' or 'boundary' characterizes type of a function; function='scaling' or 'wavelet' indicates whether the column corresponds to the values of a scaling function or a wavelet

{Compute scaling and wavelet filters using Algorithm 8 from Appendix A}
[**h**,**g**,correct_name] = Filters(namef)
{Test if a wavelet family namef is available}
if correct_name= 0 **then**
  $\mathbf{D}^W = [\,]$; $\mathbf{ind} = [\,]$; $\mathbf{col} = [\,]$; return
end if
$K = \mathrm{length}(\mathbf{h}) - 1$ {support length of ϕ}
$s = (K + \mathrm{length}(\mathbf{g}) - 1)/2$ {support length of ψ}
$r = \lceil \log(N_b - 1)/\log(2) \rceil$ {level characterizing N_b (c.f. (28)}
{Remove levels from $\mathbf{j}$ that contain no inner function}
$j_{min} = \lceil \log(s\,2^r/(N_b - 1))/\log(2)) \rceil$ {coarsest possible level}
$\mathbf{j} = \mathbf{j}(\mathbf{j} >= j_{min})$ {removing the levels smaller than j_{min}}
{Test of parameters}
$d_j = \mathrm{length}(\mathbf{j})$; $r_b = \lceil \log(1/b)/\log(2) \rceil$ {parameter r_b from $b = 1/2^{r_b}$}
if $d_j = 0$ **then**
  fprintf('no inner functions for these values of levels **j**, increase **j**')
  $\mathbf{D}^W = [\,]$; $\mathbf{ind} = [\,]$; $\mathbf{col} = [\,]$; return
else if $r < \max(\mathbf{j}) + r_b$ **then**
  fprintf('small number of points N_b for these values of **j** and b')

$\mathbf{D}^W = [\,]$; $\mathbf{ind} = [\,]$; $\mathbf{col} = [\,]$; return
end if
{Compute scaling and wavelet generators using Algorithm 1}
$[\mathbf{phi}, \mathbf{psi}]$=WaveletGen$(\mathbf{h}, \mathbf{g}, r - j(1))$
{Compute number of scaling functions at level $j(1)$}
$\mathbf{ind} = \text{zeros}(d_j + 1, 1)$
$\text{ind}(1) = Ka - 1 + \lceil (N_b - 1)\, 2^{j(1)-r}/b \rceil$ {c.f. (34)}
{Compute number of wavelets for level l}
for $l = 1 : d_j$ **do**
  $\text{ind}\,(1 + l) = sa - 1 + \lceil (N_b - 1)\, 2^{j(l)-r}/b \rceil$ {c.f. (35)}
end for
{Compute columns of $\mathbf{D}^W$ corresponding to scaling functions}
$n_f = \text{sum}(\mathbf{ind})$; $\mathbf{D}^W = \text{zeros}(N_b, n_f)$; $\mathbf{col} = \text{cell}(n_f, 1)$
$l_s = \text{length}(\mathbf{phi})$; $n_1 = Ka - 1$ {c.f. (32)}
$n_2 = \lceil (N_b - 1)\ 2^{j(1)-r}/b \rceil - \lfloor \left((N_b - 1)\ 2^{j(1)-r} - K\right)/b \rfloor - 1$ {c.f. (33)}
{Compute columns corresponding to inner scaling functions (c.f. (38)}
for $i = n_1 + 1 : \text{ind}(1) - n_2$ **do**
  $D^W(b(i - Ka)2^{r-j(1)} + 1 : b(i - Ka)\, 2^{r-j(1)} + K\, 2^{r-j(1)} + 1, i) = \mathbf{phi}$
  $\text{col}\,\{i\} = \{j(1),\, i - Ka,$ 'inner', 'scaling'$\}$
end for
{Compute columns corresponding to boundary scaling functions (c.f. (38)}
for $i = 1 : n_1$ **do**
  $D^W(1 : l_s - b\,(n_1 - i + 1)\, 2^{r-j(1)}, i) = \text{phi}((n_1 - i + 1)\, b2^{r-j(1)} + 1 : l_s)$
  $\text{col}\,\{i\} = \{j(1), -n_1 + i - 1,$ 'boundary', 'scaling'$\}$
end for
for $i = 1 : n_2$ **do**
  $p = \text{ind}(1) - n_2 + i$ {index of column}
  $D^W(b\,(p - Ka)2^{r-j(1)} + 1 : N_b, p) = \text{phi}(1 : N_b - b(p - Ka)2^{r-j(1)})$
  $\text{col}\,\{p\} = \{j(1), -n_1 + p - 1,$ 'boundary', 'scaling'$\}$
end for
{Compute columns of $\mathbf{D}^W$ corresponding to wavelets (c.f. (40))}
$n_p = \text{ind}(1)$ {number of functions on previous levels}
for $l = 1 : d_j$ **do**
  $n_1 = sa - 1$
  $k_1 = \lfloor ((N_b - 1)2^{j(l)-r} - s)/b \rfloor$, $k_2 = \lceil ((N_b - 1)2^{j(l)-r})/b) - 1 \rceil$
  $n_2 = k_2 - k_1$; $n_f = n_1 + n_2 + k_1 + 1$; $l_w = \text{length}(\mathbf{psi})$
  for $i = n_1 + 1 : n_f - n_2$ **do**
    $D^W(b(i - sa)2^{r-j(l)} + 1 : b(i - sa)2^{r-j(l)} + s2^{r-j(l)} + 1, i + n_p) = \mathbf{psi}$
    $\text{col}\,\{i + n_p\} = \{j(l), i - sa,$ 'inner', 'wavelet'$\}$
  end for
  for $i = 1 : n_1$ **do**
    $D^W(1 : l_w - b(n_1 - i + 1)2^{r-j(l)}, i + n_p) = \text{psi}((n_1 - i + 1)b2^{r-j(l)} + 1 : l_w)$
    $\text{col}\,\{i + n_p\} = \{j(l), -n_1 + i - 1,$ 'boundary', 'wavelet'$\}$
  end for
  for $i = 1 : n_2$ **do**
    $p = n_f - n_2 + i$

$D^W(b(p-sa)2^{r-j(l)}+1:N_b, p+n_p) = \mathrm{psi}(1:N_b - b(p-sa)2^{r-j(l)})$
col $\{n_p+p\} = \{j(l), -n_1+p-1$, 'boundary', 'wavelet'$\}$
end for
psi = psi(1 : 2 : length(**psi**)), $n_p = n_p + \mathrm{ind}(l+1)$
end for

The main procedure GenDict validates input parameters, generates dictionaries $\mathbf{D}^W$ and normalizes their columns.

Algorithm 3
Procedure [$\mathbf{D}^W$, **ind**, **col**]= GenDict(name,pars)

Input:

name	name of a wavelet family, for available choices see Appendix A
pars	parameters in the form pars = $\{N_b, \mathbf{j}, b\}$

Description of the parameters:

N_b	number of points
$\mathbf{j}$	vector of levels
b	translation factor $b = 2^{-r_b}$ for some integer r_b

Output:

$\mathbf{D}^W$	wavelet dictionary
ind	ind(1) is the number of scaling functions at level $j(1)$, and ind(k) for $k > 1$ is the number of wavelets at level $j(k-1)$
col	cell array such that col $\{n\} = \{j, k, \mathrm{type}, \mathrm{function}\}$, if the n-th column of $\mathbf{D}^W$ corresponds to values of scaling function $\phi(2^j x - bk)$ or wavelet $\psi(2^j x - bk)$; type='inner' or 'boundary' characterizes type of a function; function='scaling' or 'wavelet' indicates whether the column corresponds to the values of a scaling function or a wavelet

{Define cell array of names of all available families}
families= { 'CW2','CW3','CW4','CDF97','CDF97d','CDF53', 'Short4','Short3', 'Short2', 'Db3','Db4','Db5','Sym3','Sym4', 'Sym5','Coif26','Coif38' }
{Validate input parameters}
if nargin $\neq$ 2 **then**
error('Need 2 input arguments')
end if
if ~ischar(namef) **then**
error('Name must be a string')
end if
N_b = pars $\{1\}$; $\mathbf{j}$ = pars $\{2\}$; b = pars $\{3\}$; $\mathbf{j}$ = sort($\mathbf{j}$)
if $b \leq 0$ **then**

error('I expect $b > 0$')
end if
$r = \log(1/b)/\log(2)$
if $|r - \text{round}(r)| > 10^{-10}$ **then**
fprintf('Choose b such that $1/b = 2^r$ for some integer r')
$\mathbf{D}^W = [\,]$; $\mathbf{ind} = [\,]$; $\mathbf{col} = [\,]$; return
else if ismember({namef},families) **then**
{Generate dictionary using Algorithm 2}
$[\mathbf{D}^W, \mathbf{ind}, \mathbf{col}] =$ WaveletDict(namef, N_b, $\mathbf{j}$, b)
{Normalize columns of $\mathbf{D}^W$ using Algorithm 9 from Appendix A}
$\mathbf{D}^W =$ NormDict($\mathbf{D}^W$,1)
else
error('Unknown name of a wavelet family')
end if

Remark 2. It is worth remarking that the range of scales, say $\mathbf{j} = (j_0, \ldots, J)$ depends on length of the signal partition. For a signal segment of length $N_b = 2^r + 1$ a dictionary contains values of scaling functions and wavelets at points $l/2^r$ for some integer r. For a signal segment of length $2N_b - 1 = 2^{r+1} + 1$ a dictionary contains values of functions at points $l/2^{r+1}$. Thus we have

$$\phi_{j,k,b}\left(\frac{l}{2^r}\right) = \phi\left(2^j \frac{l}{2^r} - kb\right) = \phi\left(2^{j+1}\frac{l}{2^{r+1}} - kb\right) = \phi_{j+1,k,b}\left(\frac{l}{2^{r+1}}\right) \quad (41)$$

and

$$\psi_{j,k,b}\left(\frac{l}{2^r}\right) = \psi\left(2^j \frac{l}{2^r} - kb\right) = \psi\left(2^{j+1}\frac{l}{2^{r+1}} - kb\right) = \psi_{j+1,k,b}\left(\frac{l}{2^{r+1}}\right). \quad (42)$$

Therefore, nonzero elements of vectors on the level j in a dictionary for $\mathbb{R}^{N_b}$ correspond to nonzero elements of vectors on the level $j+1$ in a dictionary for $\mathbb{R}^{2N_b-1}$. This situation is illustrated in Figure 2, where vectors of values $\phi_{j,0,b}\left(l/2^r\right)$ are displayed for $j = 2$ and $r = 4$ and for $j = 3$ and $r = 5$. Note that the nonzero elements in these vectors are the same. Therefore, if for the signal segment of length N_b the vector $\mathbf{j} = (j_0, \ldots, J)$ is used, then we recommend to use the vector $\mathbf{j} = (j_0, \ldots, J+1)$ for the signal segment of length $2N_b - 1$, and similarly to use levels $\mathbf{j} = (j_0, \ldots, J+m)$ for the signal segment of length $2^m N_b - 1$.

Example 1. To build dictionaries for the wavelet family 'Short3' at levels 2 and 3, for translation parameter $b = 1/4$, and the number of points $N_b = 33$, use the procedure TestDict below.

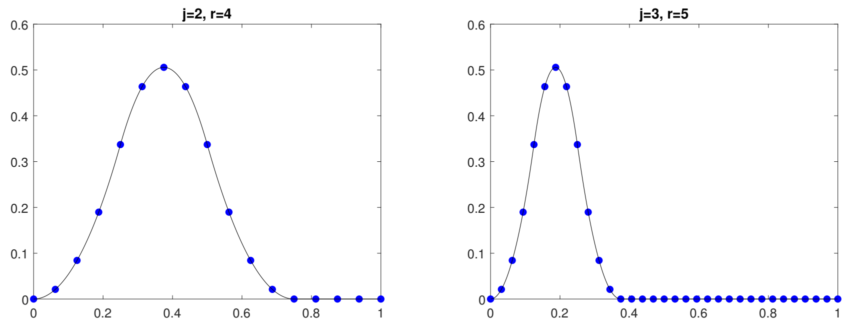

Figure 2: Vectors of values $\phi_{j,0,b}\left(l/2^r\right)$ for $j=2$ and $r=4$ (left) and for $j=3$ and $r=5$ (right).

Procedure TestDict

---

namef='Short3'; $N_b = 33$; $\mathbf{j} = 2:3$; $b = 1/4$
$[\mathbf{D}^W, \mathbf{ind}, \mathbf{col}]$=GenDict(namef,$\{N_b, \mathbf{j}, b\}$)

The output is the matrix $\mathbf{D}^W$ of size 33×97 and the vector $\mathbf{ind} = [27, 27, 43]$. This means that there are 27 scaling functions at level 2, 27 wavelets at level 2, and 43 wavelets at level 3. The cell array **col** characterizes functions corresponding to columns of $\mathbf{D}^W$. For example

$$\mathrm{col}\,\{30\} = \{2, -9, \text{`boundary'}, \text{`wavelet'}\} \tag{43}$$

which means that 30th column of the matrix $\mathbf{D}^W$ contains values of a wavelet function $\psi\left(2^2 x - b\,(-9)\right)$. This wavelet is a boundary wavelet, i.e., only a part of its support lies in the interval I defined by (30). Some of the vectors from this dictionary corresponding to values of scaling functions are displayed in Figure 3 and some of the vectors corresponding to values of wavelets are displayed in Figure 4.

2.4. Construction of dictionaries for ECG modelling

As mentioned in Sec. 1, because ECG signals are usually superimposed to a baseline or smooth background, the full dictionary $\mathbf{D}$ we use for ECG modelling is built as follows

$$\mathbf{D} = [\mathbf{D}^C \, \mathbf{D}^W], \tag{44}$$

where $\mathbf{D}^W$ is the output of Algorithm 5 and $\mathbf{D}^C$ is a matrix containing a few low frequency components from a discrete cosine basis. Before normalization $\mathbf{D}^C$ is given as

$$D^C(k,n) = \cos(\pi(2k-1)(n-1)/(2N_b)),\; k = 1,\ldots,N_b,\; n = 1,\ldots,M_c, \tag{45}$$

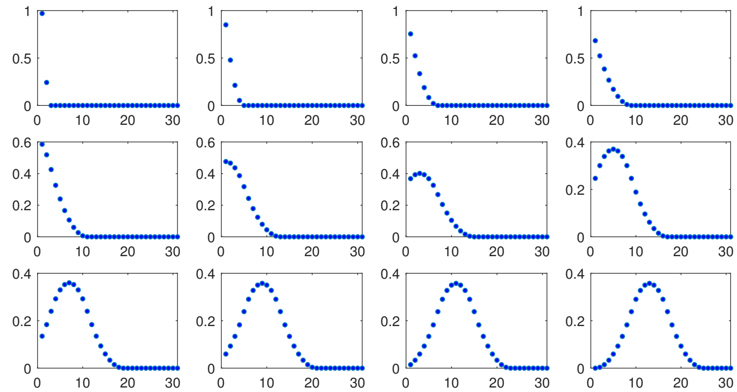

Figure 3: Plots of 12 vectors from the dictionary $\mathbf{D}^W$ from Example 1 corresponding to scaling functions on the level 2.

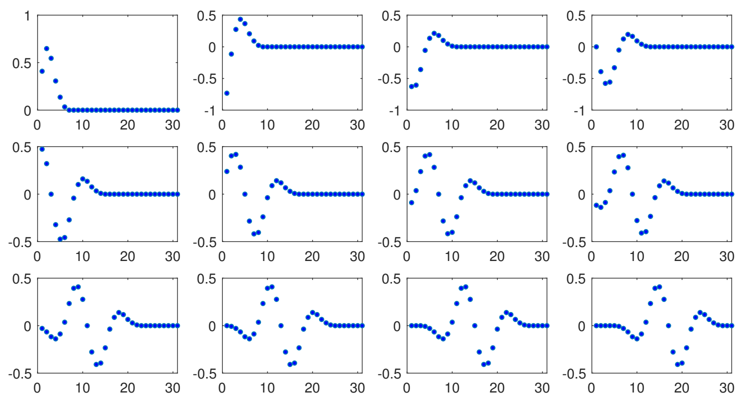

Figure 4: Plots of 12 vectors from the dictionary $\mathbf{D}^W$ from Example 1 corresponding to wavelets on the level 2.

where M_c is a small number in comparison to N_b. For the numerical examples of the next section we consider $M_c = 10$. Algorithm 5 computes $\mathbf{D}^C$.

Algorithm 5
Procedure $\mathbf{D}^C = \text{DCos}(N_b, M_c)$

Input:

N_b	the size of the Euclidean space the vectors should belong to
M_c	number of frequencies to use starting from 0

Output:

$\mathbf{D}^C$	matrix whos columns are discrete cosine vectors

$n = 1 : M_c$; $k = 1 : N_b$
$\mathbf{D}^C = \cos(\pi(2k-1)^T(n-1)/(2N_b))$
$\mathbf{D}^C$ = NormDict($\mathbf{D}^C$,1)

---

3. Method for construction of the model

In this section we present the procedures for constructing the ECG signal model (c.f. Algorithm 6) and for calculating the assessment metrics. The quality of the signal approximation is assessed with respect to the PRD defined as follows

$$\mathrm{PRD} = \frac{\|\mathbf{f} - \mathbf{f}^{\mathrm{r}}\|}{\|\mathbf{f}\|} \times 100\%, \tag{46}$$

where $\mathbf{f}$ is the original signal and $\mathbf{f}^{\mathrm{r}}$ is the signal reconstructed by concatenation of the approximated segments $\mathbf{f}^{\mathrm{a}}\{q\}$, $q = 1, \ldots, Q$.

The local PRD with respect to every segment in the signal partition is indicated as $\mathrm{prd}(q)$, $q = 1, \ldots, Q$ and calculated as

$$\mathrm{prd}(q) = \frac{\|\mathbf{f}\{q\} - \mathbf{f}^{\mathrm{a}}\{q\}\|}{\|\mathbf{f}\{q\}\|} \times 100\%, \quad q = 1, \ldots, Q. \tag{47}$$

For the signal approximation the OOMP method is stopped through a fixed value tol so as to achieve the same value of prd for all the segments in the records. Assuming that the target prd before quantization is prd_0 we set $\mathrm{tol} = \mathrm{prd}_0\|\mathbf{f}_q\|/100$.

The goal of the signal model is to approximate each segment in the signal partition using as few atoms as possible. Thus, for a fixed value of PRD, the sparsity of the signal representation is assessed by the sparsity ratio (SR)

$$\mathrm{SR} = \frac{N}{K}, \tag{48}$$

where N is the total length of the signal and $K = \sum_{q=1}^{Q} k(q)$, with $k(q)$ the number of atoms in the atomic decomposition (1) of each segment of length N_b. The corresponding quantity evaluated for every cell in the partition is the local sparsity ratio

$$\mathrm{sr}(q) = \frac{N_b}{k(q)}, \quad q = 1, \ldots, Q. \tag{49}$$

This local quantity is relevant to the detection of non-stationary noise, significant distortion in ECG patterns, or changes of morphology in the heart beats.

Given an ECG signal $\mathbf{f}$ the procedure described in Algorithm 6 constructs the signal approximation, $\mathbf{f}^{\mathrm{r}}$, using the dictionaries introduced in the previous section.

Algorithm 6
Procedure $[\mathbf{f}^{\mathrm{r}}, \boldsymbol{\ell}, \mathbf{c}, \mathrm{prd}, \mathrm{sr}, \mathrm{PRD}, \mathrm{SR}]$= SignalModel$(\mathbf{f}, N_b, \mathrm{prd}_0, \mathrm{namef}, \mathrm{pars}, M_c)$

Input:

$\mathbf{f}$	signal
N_b	number of points in each segment of the partition
prd_0	parameter to control the approximation error
namef	name of a wavelet family
pars	parameters as described in Algorithm 3
M_c	number M_c of components in the cosine subdictionary

Output:

$\mathbf{f}^{\mathrm{r}}$	approximated signal
$\boldsymbol{\ell}$	cell with the indices of the atoms in the atomic decomposition of each element in the partition
$\mathbf{c}$	cell with the coefficients in the atomic decomposition of each element in the partition
prd	vector $\mathrm{prd} \in \mathbb{R}^Q$ (cf. (47))
sr	vector $\mathrm{sr} \in \mathbb{R}^Q$ (cf. (49))
PRD	global PRD
SR	global SR

{Create the signal partition using Algorithm 10}
$[\mathbf{f}^{\mathrm{c}}, Q, \mathbf{f}]$=Partition$(\mathbf{f}, N_b)$
{Construct the wavelet dictionary $\mathbf{D}^W$ using Algorithm 3 given in Appendix B}
$[\mathbf{D}^W$, ind]= GenDict(namef, pars)
{Construct the component $\mathbf{D}^C$ using Algorithm 5 given in Appendix B}
$[\mathbf{D}^C]$=DCos(N_b, M_c)
{Merge $\mathbf{D}^C$ and $\mathbf{D}^W$ to create dictionary $\mathbf{D}$}
$\mathbf{D} = [\mathbf{D}^C\ \mathbf{D}^W]$
Set $\mathbf{f}^{\mathrm{r}} = [\,]$, $K = 0$ and N =length$(\mathbf{f})$.
for q=1:Q **do**
  tol = $\mathrm{prd}_0 \|\mathbf{f}^{\mathrm{c}}\{q\}\|/100$
  {Call the OOMP function to construct the model (c.f. (1))}
  $[\mathbf{f}^{\mathrm{a}}\{q\}, \boldsymbol{\ell}\{q\}, \mathbf{c}\{q\}]$=OOMP$(\mathbf{f}^{\mathrm{c}}\{q\}, \mathbf{D}, \mathrm{tol}, 1)$
  {Calculate local sr and prd (c.f. (49) and (47)}
  $\mathrm{prd}(q) = \frac{\|\mathbf{f}^{\mathrm{c}}\{q\} - \mathbf{f}^{\mathrm{a}}\{q\}\|}{\|\mathbf{f}^{\mathrm{c}}\{q\}\|} \times 100$
  $k(q) = \mathrm{length}(\mathbf{c}\{q\}))$
  $\mathrm{sr}(q) = N_b/k(q)$
  $K = K + k(q)$
  $\mathbf{f}^{\mathrm{r}} = [\mathbf{f}^{\mathrm{r}}\ \mathbf{f}^{\mathrm{a}}\{q\}]$
end for
{Calculate global SR and PRD (c.f. (48) and (46))}
$\mathrm{SR} = N/K$; $\mathrm{PRD} = \frac{\|\mathbf{f} - \mathbf{f}^{\mathrm{r}}\|}{\|\mathbf{f}\|} \times 100$

4. Numerical results and discussion

We illustrate now the use of the software to approximate records 117, 202, and 231 in the MIT-BIH Arrhythmia database. Each record consists of 650000 samples and is partitioned for the approximation in segments of $N_b = 500$ points each. Table 1 gives the values of the SR (c.f. (48)) achieved using wavelet bases, denoted as $\mathrm{SR_B}$, and wavelet dictionaries denoted as $\mathrm{SR_D}$. The wavelet families are indicated in the first column of Table 1. The wavelet dictionary is constructed with scales $\mathbf{j} = (3, \ldots, 7)$ and translation parameter $b = 1/4$, whilst the wavelet basis entails to add one more scale and a translation parameter $b = 1$. In all the cases the approximation is realized to obtain PRD = 0.51%.

Table 1 is produced by running the script 'Run_ECG_Appox' and changing the variable 'namef' to the corresponding family option.

Procedure Run_ECG_Approx

{Read the signal $\mathbf{f}$}
file='Record_231_11bits.dat'
fid=fopen(file,'r')
$\mathbf{f}$=fread(fid,'ubit11')
fclose(fid)
{Set the required PRD for the approximation}
$\mathrm{prd}_0 = 0.53$
{Set the length for each segment in the signal partition}
$N_b = 500$
{Set the parameters for the wavelet dictionary}
namef ='CDF97'; $b = 0.25$; $\mathbf{j} = 3 : 7$; pars = $\{N_b, \mathbf{j}, b\}$
{Set the number of cosine components}
$M_c = 10$
{Construct the signal module}
$[\mathbf{f}^{\mathrm{r}}, \boldsymbol{\ell}, \mathbf{c}, \mathrm{prd}, \mathrm{sr}, \mathrm{PRD}, \mathrm{SR}] = \mathrm{SignalModel}(\mathbf{f}, N_b, \mathrm{prd}_0, \mathrm{namef}, \mathrm{pars}, M_c)$
{Plote the first 2000 sample points in the signal, the approximation and the error}

As observed in the Table 1, the gain in dimensionality reduction (larger value of SR) is significant when consider a wavelet dictionary, instead of a wavelet basis, as for constructing the component $\mathbf{D}^W$ of the full dictionary. This results were demonstrated in [29] on the whole MIT-BIH Arrhythmia database, which motivated this Communication to provide the details and algorithm for the actual construction of wavelet dictionaries from different options for the wavelet prototypes.

In our algorithm, we use the OOMP method for the selection of atoms because we have found that the OOMP method produces superior results to other greedy techniques, such as Matching Pursuit [24] and Orthogonal Matching Pursuit [27], and also superior results to the Basis Pursuits methods based on

Table 1: SRs achieved using dictionaries with $\mathbf{D}^W$ component as indicated in the first column of the table. SR_B are values of SR obtained if $\mathbf{D}^W$ is a basis and SR_D if $\mathbf{D}^W$ is a dictionary.

Rec.	117		202		231	
$\mathbf{D}^W$	SR_B	SR_D	SR_B	SR_D	SR_B	SR_D
CW2	17.5	26.5	17.3	24.5	15.7	23.0
CW3	17.4	28.1	15.9	24.9	15.6	24.0
CW4	15.7	24.8	14.3	22.5	18.4	21.9
CDF97	21.5	30.3	21.4	28.4	19.5	27.5
CDF97d	17.2	23.5	17.3	22.5	15.8	21.9
CDF53	22.4	29.6	23.6	27.0	20.2	27.0
Db3	18.5	23.7	18.1	22.7	16.6	22.7
Db4	19.0	25.7	19.1	24.7	17.7	24.1
Db5	20.4	26.1	18.7	24.2	17.8	24.1
Sym3	18.4	23.8	18.1	22.7	16.6	22.7
Sym4	19.7	27.5	19.5	25.8	17.7	25.1
Sym5	20.5	28.3	20.6	28.5	18.4	25.4
Short2	8.2	27.9	8.7	26.3	8.1	24.7
Short3	19.6	31.8	18.3	27.6	17.8	27.3
Short4	9.5	29.1	10.1	27.6	9.1	26.6
Coif26	17.7	23.0	17.7	21.8	16.3	24.7
Coif38	19.5	28.5	19.7	26.5	17.8	26.1

1-norm minimization [10], any of these techniques, or others, can be applied with the identical dictionaries. The aim of this paper was to produce details for the construction of the wavelet dictionaries given rise to piecewise modeling of ECG signals using any suitable approach for the selection process.

The top left graph in Figure 5 illustrates the first 2000 points in the record 231 and the approximation for PRD = 0.51%. The top right graph represents the values of local sparsity $1/\text{sr}(q)$, $q = 1, \dots, 1300$ for the same record. It is noticed that these values can be classified into two well defined bands. The bottom left graph in Figure 5 shows a typical heart beat in a frame corresponding to a value $1/\text{sr}$ in the upper band, and the bottom right graph to a value in the lower band. The morphologic difference between the two heart beats is noticeable at a glance.

5. Conclusions

Details on the construction of wavelets dictionaries for modeling ECG signals have been provided. The use of the software, which has been made publicly available on a dedicated website [15], was illustrated to reduce the dimensionality of three records from the MIT-BIH Arrhythmia database. The conclusions coincide with those that were drawn in the previous publication [29] using the

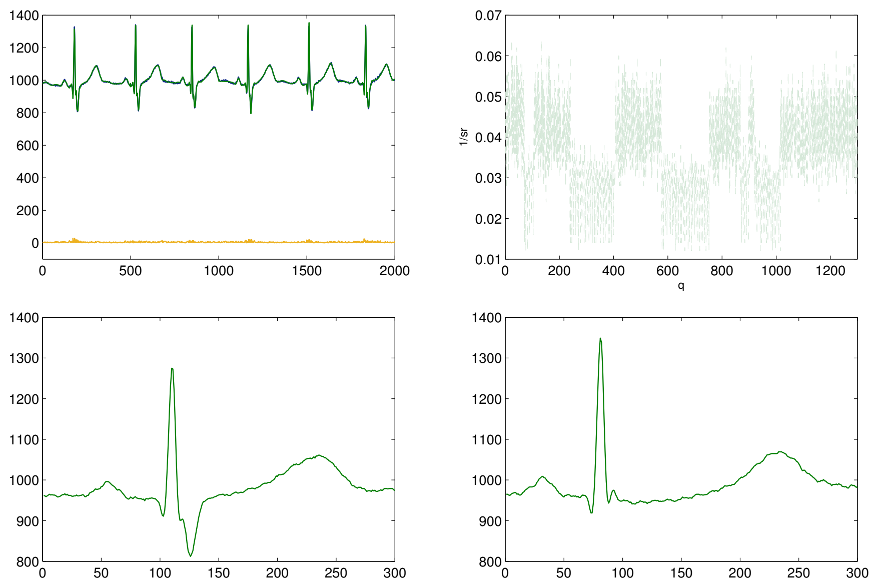

Figure 5: The waveforms in the top left graph are the raw data and the approximations corresponding to 2000 points in the records 231 (the bottom line in the same graph is the point-wise error). The top right graph plots the values 1/sr for records 231. The bottom left graph is a typical heart beat in a segment for which the values of $1/\mathrm{sr}(q)$ belongs to the upper band. In the bottom right graph the heart beat corresponds to a frame in the lower band.

whole database. However, regardless of the particular application, the purpose of this paper was to provide a complete description of the construction of the wavelets dictionaries, which had not been addressed in [29]. We believe the proposed dictionaries should be of assistance to general applications which relay on dimensionality reduction at low level distortion as a first step of further ECG signal processing.

Appendix A

In this appendix, we present auxiliary procedures used in algorithms in Section 2. In the algorithms, 'namef' denotes a name of a wavelet family, available choices are:

namef='CW2'	Chui-Wang linear spline wavelets [11]
='CW3'	Chui-Wang quadratic spline wavelets [11]
='CW4'	Chui-Wang cubic spline wavelets [11]
='CDF97'	primal CDF97 wavelets [6]
='CDF97d'	dual CDF97 wavelets [6]
='CDF53'	primal CDF53 wavelets [6]

=‘Short4’	cubic spline wavelet with short support and 4 vanishing moments [9, 18]
=‘Short3’	quadratic spline wavelet with short support and 3 vanishing moments [9, 18]
=‘Short2’	linear spline wavelet with short support and 2 vanishing moments [9, 18]
=‘Db3’	Daubechies wavelet with 3 vanishing moments [16]
=‘Db4’	Daubechies wavelet with 4 vanishing moments [16]
=‘Db5’	Daubechies wavelet with 5 vanishing moments [16]
=‘Sym3’	symlet with 3 vanishing moments [17]
=‘Sym4’	symlet with 4 vanishing moments [17]
=‘Sym5’	symlet with 5 vanishing moments [17]
=‘Coif26’	coiflet with 2 vanishing moments and the support length 6 that is most regular [17]
=‘Coif38’	coiflet with 3 vanishing moments and the support length 8 that is most symmetrical [17]

A wavelet basis is determined by its scaling and wavelet filters. Algorithm 8 assigns these filters for a chosen wavelet family, the values of filters are computed by methods from [6, 9, 11, 12, 16, 17, 18].

Algorithm 8
Procedure [**h**,**g**,correct_name] = Filters(namef)

Input:

namef	name of a wavelet family

Output:

h	scaling filter for a wavelet family specified by ‘namef’
g	wavelet filter for a wavelet family specified by ‘namef’
correct_name	returns 1 if ‘namef’ is a name of an available wavelet family, otherwise returns 0

correct_name=1
switch (namef)
case ‘CW2’**:**
$\mathbf{h} = [1/2, 1, 1/2]$; $\mathbf{g} = [1, -6, 10, -6, 1]/12$
case ‘CW3’ **:**
$\mathbf{h} = [1/4, 3/4, 3/4, 1/4]$; $\mathbf{g} = [1, -29, 147, -303, 303, -147, 29, -1]/480$
case ‘CW4’ **:**
$\mathbf{h} = [1/8, 1/2, 3/4, 1/2, 1/8]$
$\mathbf{g} = [1, -124, 1677, -7904, 18482, -24264, 18482, -7904, 1677, -124, 1]/2520$
case ‘CDF97’**:**
$\mathbf{h} = [-0.045635881557, -0.028771763114, 0.295635881557, 0.557543526229, 0.295635881557, -0.028771763114, -0.045635881557]$
$\mathbf{g} = [0.026748757411, 0.016864118443, -0.078223266529, -0.266864118443\ 0.602949018236, -0.266864118443, -0.078223266529, 0.016864118443, 0.026748757411]$

case 'CDF97d'**:**
$\mathbf{h} = [0.026748757411, -0.016864118443, -0.078223266529, 0.266864118443, 0.602949018236, 0.266864118443, -0.078223266529, -0.016864118443, 0.026748757411000]$
$\mathbf{g} = [0.045635881557, -0.028771763114, -0.295635881557, 0.557543526229, -0.295635881557, -0.028771763114, 0.045635881557]$
case 'CDF53'**:**
$\mathbf{h} = [1/2, 1, 1/2]$; $\mathbf{g} = [-1/8, -1/4, 3/4, -1/4, -1/8]$
case 'Short4'**:**
$\mathbf{h} = [1/8, 1/2, 3/4, 1/2, 1/8]$; $\mathbf{g} = [1/8, -1/2, 3/4, -1/2, 1/8]$
case 'Short3'**:**
$\mathbf{h} = [1/4, 3/4, 3/4, 1/4]$; $\mathbf{g} = [-1/4, 3/4, -3/4, 1/4]$
case 'Short2'**:**
$\mathbf{h} = [1/2, 1, 1/2]$; $\mathbf{g} = [-1/2, 1, -1/2]$
case 'Db3' **:**
$\mathbf{h} = [0.035226291882101, -0.085441273882241, -0.135011020010391, 0.459877502119331, 0.806891509313339, 0.332670552950957]$
$\mathbf{g} = [-0.332670552950957, 0.806891509313339, -0.459877502119331, -0.135011020010391, 0.085441273882241, 0.035226291882101]$
case 'Db4'**:**
$\mathbf{h} = [0.162901714025620, 0.505472857545650, 0.446100069123190, -0.019787513117910, -0.132253583684370, 0.021808150237390, 0.023251800535560, -0.007493494665130]$
$\mathbf{g} = -\text{fliplr}([0.162901714025620, -0.505472857545650, 0.446100069123190, 0.019787513117910, -0.132253583684370, -0.021808150237390, 0.023251800535560, 0.007493494665130])$
case 'Db5'**:**
$\mathbf{h} = [0.003335725285002, -0.012580751999016, -0.006241490213012, 0.077571493840065, -0.032244869585030, -0.242294887066190, 0.138428145901103, 0.724308528438574, 0.603829269797473, 0.160102397974125]$
$\mathbf{g} = [-0.160102397974125, 0.603829269797473, -0.724308528438574, 0.138428145901103, 0.242294887066190, -0.032244869585030, -0.077571493840065, -0.006241490213012, 0.012580751999016, 0.003335725285002]$
case 'Sym3'**:**
$\mathbf{h} = [0.035226291882101, -0.085441273882241, -0.135011020010391, 0.459877502119331, 0.806891509313339, 0.332670552950957]$
$\mathbf{g} = [-0.332670552950957, 0.806891509313339, -0.459877502119331, -0.135011020010391, 0.085441273882241, 0.035226291882101]$
case 'Sym4'**:**
$\mathbf{h} = [0.022785172948000, -0.008912350720850, -0.070158812089500, 0.210617267102000, 0.568329121705000, 0.351869534328000, -0.020955482562550, -0.053574450709000]$
$\mathbf{g} = \text{fliplr}([0.022785172948000, 0.008912350720850, -0.070158812089500, -0.210617267102000, 0.568329121705000, -0.351869534328000,$

$-0.020955482562550, 0.053574450709000])$
case 'Sym5' **:**
$\mathbf{h} = [0.027333068345078, 0.029519490925775, -0.039134249302383, 0.199397533977394, 0.723407690402421, 0.633978963458212, 0.016602105764522, -0.175328089908450, -0.021101834024759, 0.019538882735287]$
$\mathbf{g} = [-0.019538882735287, -0.021101834024759, 0.175328089908450, 0.016602105764522, -0.633978963458212, 0.723407690402421, -0.199397533977394, -0.039134249302383, -0.029519490925775, 0.027333068345078]$
case 'Coif26'**:**
$\mathbf{h} = [9 - \sqrt(15), 13 + \sqrt(15), 6 + 2\sqrt(15), 6 - 2\sqrt(15), 1 - \sqrt(15), -3 + \sqrt(15)]/32$
$\mathbf{g} = -\text{fliplr}([9 - \sqrt(15), -13 - \sqrt(15), 6 + 2\sqrt(15), -6 + 2\sqrt(15), 1 - \sqrt(15), 3 - \sqrt(15)]/32)$
case 'Coif38'**:**
$\mathbf{h} = [-1/32 - \sqrt(7)/128, -3/128, 9/32 + 3\sqrt(7)/128, 73/128, 9/32 - 3\sqrt(7)/128, -9/128, -1/32 + \sqrt(7)/128, 3/128]$
$\mathbf{g} = -\text{fliplr}([-1/32 - \sqrt(7)/128, 3/128, 9/32 + 3\sqrt(7)/128, -73/128, 9/32 - 3\sqrt(7)/128, 9/128, -1/32 + \sqrt(7)/128, -3/128])$
otherwise
disp('wrong name of a wavelet family')
correct_name = 0
end switch

Now, we introduce a simple procedure NormDict for normalization of dictionaries. More precisely, this procedure normalizes the columns of dictionary $\mathbf{D}$ to have the Euclidean norm equaled to $1/\sqrt{\delta}$.

Algorithm 9
Procedure $\mathbf{D}$ = NormDict($\mathbf{D}$, δ)

Input:

$\mathbf{D}$	wavelet dictionary
δ	parameter such that prescribed norm size is $1/\sqrt{\delta}$

Output:

$\mathbf{D}$	normalized wavelet dictionary such that the Euclidean norm of each column is $1/\sqrt{\delta}$

tol=10^{-5}
if nargin=1 **then**
$\delta = 1$
end if
N=size($\mathbf{D}$,2); $i = 0$
while $i < N$ **do**

$i = i + 1$; nor $= \sqrt{\delta}\ \|D(:, i)\|$
if nor > tol **then**
$D(:, i) = D(:, i)/\text{nor}$
else
$D(:, i) = [\,]; N = N - 1$
end if
end while

Appendix B

In this appendix, we present auxiliary procedures used in algorithms in Section 3. The next procedure Partition creates a partition of the signal $\mathbf{f}$ into Q segments of the prescribed length N_b.

Algorithm 10
Procedure $[\mathbf{f}^{\mathrm{c}}, Q, \mathbf{f}]$=Partition$(\mathbf{f}, N_b)$

Input:

$\mathbf{f}$	signal
N_b	length of each segment in the partition

Output:

$\mathbf{f}^{\mathrm{c}}$	cells $\mathbf{f}^{\mathrm{c}}\{q\}$, $q = 1, \ldots, Q$ with the signal partition
Q	number of cells in the partition
$\mathbf{f}$	resized signal to be of length QN_b

$N = \text{length}(\mathbf{f}); Q = \lfloor \frac{N}{N_b} \rfloor; t_o = 1$
$\mathbf{f} \leftarrow f(1 : QN_b)$
for $q = 1 : Q$ **do**
$t = t_o : t_o + N_b - 1;\ t_0 = t_0 + N_b$
$\mathbf{f}^{\mathrm{c}}\{q\} = f(t)$
end for

The procedure for signal approximation using OOMP method is presented below.

Algorithm 11
Procedure $[\mathbf{f}^{\mathrm{a}}, \boldsymbol{\ell}, \mathbf{c}]$= OOMP$(\mathbf{f}, \mathbf{D}, \text{tol}, \ell_1)$

Input:

$\mathbf{f}$	signal to be approximated by an atomic decomposition
$\mathbf{D}$	wavelet dictionary
tol	parameter to control the approximation error
ℓ_1	index of the atom for initializing the OOMP algorithm

Output:

$\mathbf{f}^{\mathrm{a}}$	approximation of the signal $\mathbf{f}$ (c.f. (1))
$\boldsymbol{\ell}$	vector whose components are the indices of the selected columns from the input dictionary
$\mathbf{c}$	coefficients $\mathbf{c} \in \mathbb{R}^{N_b}$ of the atomic decomposition (c.f. (1))

{The method implemented in this function is fully described in the main paper [29].}

References

[1] U.R. Acharya, H. Fujita, M. Adam, S.L. Oh, K.V. Sudarshan, J.H. Tan, J.E.W. Koh, Y. Hagiwara, C.K. Chua, C.K. Poo, R.S. Tan, Automated characterization and classification of coronary artery disease and myocardial infarction by decomposition of ECG signals: A comparative study, Information Sciences 377 (2017) 17–29. doi:https://doi.org/10.1016/j.ins.2016.10.013

[2] A. Adamo, G. Grossi, R. Lanzarotti, J. Lin, ECG compression retaining the best natural basis k-coefficients via sparse decomposition, Biomedical Signal Processing and Control 15 (2015) 11–17. doi:https://doi.org/10.1016/j.bspc.2014.09.002

[3] M. Andrle, L. Rebollo-Neira, Cardinal B-spline dictionaries on a compact interval, Applied and Computational Harmonic Analysis 18 (2005) 336–346. doi:10.1016/j.acha.2005.01.001.

[4] M. Andrle and L. Rebollo-Neira, From cardinal spline wavelet bases to highly coherent dictionaries, Journal of Physics A 41 (2008), article No. 172001. doi:10.1088/1751-8113/41/17/172001.

[5] A.Y. Hannun, P. Rajpurkar, M. Haghpanahi, G.H. Tison, C. Bourn, M.P. Turakhia, A.Y. Ng, Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network, Nature Medicine 25 (2019) 65–69. doi:https://doi.org/10.1038/s41591-018-0268-3

[6] A. Cohen, I. Daubechies, J.C. Feauveau, Biorthogonal bases of compactly supported wavelets, Communications on Pure and Applied Mathematics 45 (1992) 485–560. doi:10.1002/cpa.3160450502.

[7] V.H.C. de Albuquerque, T.M. Nunes, D.R. Pereira, E. J. da Luz, D. Menotti, J. P. Papa, J. M. R. S. Tavareas, Robust automated cardiac arrhythmia detection in ECG beat signals, Neural Computing and Applliccations 29 (2018) 679–693. doi:https://doi.org/10.1007/s00521-016-2472-8.

[8] A. Cohen, Numerical Analysis of Wavelet methods, Studies in Mathematics and its Applications 32, Elsevier, Amsterdam, 2003.

[9] D. Chen, Spline wavelets of small support, SIAM Journal on Mathematical Analysis 26 (1995) 500–517. doi:10.1137/S0036141093245264.

[10] S.S. Chen, D.L. Donoho, M.A. Saunders, Atomic decomposition by basis pursuit, SIAM Journal on Scientific Computing 20 (1998) 33–61. doi:10.1137/S1064827596304010

[11] C. Chui, J. Wang, On compactly supported spline wavelets and a duality principle, Transactions of the American Mathematical Society 330 (1992) 903–915. doi:10.2307/2153941.

[12] D. Černá, V. Finěk, K. Najzar, On the exact values of coefficients of coiflets, Central European Journal of Mathematics 6 (2008) 159–169. doi:10.2478/s11533-008-0011-2.

[13] `https://physionet.org/physiobank/database/mitdb/` (Last access July 2019).

[14] `http://www.nonlinear-approx.info/examples/node011.html` (Last access July 2019).

[15] `http://www.nonlinear-approx.info/examples/node013.html` (Last access July 2019).

[16] I. Daubechies, Orthonormal bases of compactly supported wavelets, Communications on Pure and Applied Mathematics 41 (1988) 909-996. doi:10.1002/cpa.3160410705

[17] I. Daubechies, Orthonormal bases of compactly supported wavelets II, variations on a theme, SIAM Journal on Mathematical Analysis 24 (1993) 499-519. doi:10.1137/0524031.

[18] B. Han, Z. Shen, Wavelets with short support, SIAM Journal on Mathematical Analysis 38 (2006) 530–556. doi:10.1137/S0036141003438374.

[19] S.J. Lee, J. Kim, M. Lee, A real-time ECG data compression and transmission algorithm for an e-health device, IEEE Transactions on Biomedical Engineering 58 (2011) 2448–2455. doi:10.1109/TBME.2011.2156794.

[20] H. Li, D. Yuan, X. Ma, D. Cui, L. Cao, Genetic algorithm for the optimization of features and neural networks in ECG signals classification, Scientific Reports 7 (2017), Article No. 41011. doi:10.1038/srep41011

[21] E. J. Luz, W. Robson Schwartz, G. Camara Chavez, D. Mennotti, ECG-based heartbeat classification for arrhythmia detection: A survey, Computer Methods and Programs in Biomedicine 127 (2016) 144–164. doi: https://doi.org/10.1016/j.cmpb.2015.12.008

[22] A. Lyon , A. Mincholé, J. P. Martínez, P. Laguna, B. Rodriguez, Computational techniques for ECG analysis and interpretation in light of their contribution to medical advances, Journal of the Royal Society of Interface 15 (2018), article No. 20170821 https://doi.org/10.1098/rsif.2017.0821.

[23] J.L. Ma, T.T. Zhang, M. C. Dong, A novel ECG data compression method using adaptive Fourier decomposition with security guarantee in e-health applications, IEEE Journal of Biomedical and Health Informatics, 19 (2015) 986–994. doi:10.1109/JBHI.2014.2357841.

[24] S. G. Mallat, Z. Zhang, Matching Pursuits with Time-Frequency Dictionaries, IEEE Transactions on Signal Processing 41 (1993) 3397–3415. doi:10.1109/78.258082

[25] A. Pandey, B. Singh, S Sain, N. Sood, A joint application of optimal threshold based discrete cosine transform and ASCII encoding for ECG data compression with its inherent encryption, Australasian Physical and Engineering Sciences in Medicine 39 (2016) 833–855. doi:10.1007/s13246-016-0476-4.

[26] P. Pławiak, Novel methodology of cardiac health recognition based on ECG signals and evolutionary-neural system, Expert Systems with Applications 92 (2018) 334–349. doi:https://doi.org/10.1016/j.eswa.2017.09.022

[27] Y.C. Pati, R. Rezaiifar, P.S. Krishnaprasad, Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition, Proceedings of the 27th Annual Asilomar Conference in Signals, System and Computers, vol 1, (1993) 40–44. doi:10.1109/ACSSC.1993.342465

[28] S. Raj, K.C. Ray, Sparse representation of ECG signals for automated recognition of cardiac arrhythmias, Expert Systems with Applications 105 (2018) 49–64. doi:https://doi.org/10.1016/j.eswa.2018.03.038

[29] L. Rebollo-Neira, D. Černá, Wavelet based dictionaries for dimensionality reduction of ECG signals, Biomedical Signal Processing and Control 54 (2019), article No. 101593. doi:https://doi.org/10.1016/j.bspc.2019.101593.

[30] L. Rebollo-Neira, D. Lowe, Optimized orthogonal matching pursuit approach, IEEE Signal Process. Letters 9 (2002) 137–140. doi:10.1109/LSP.2002.1001652.

[31] L. Rebollo-Neira, Cooperative greedy pursuit strategies for sparse signal representation by partitioning, Signal Processing 125 (2016) 365–375. doi:10.1016/j.sigpro.2016.02.008.

[32] L. Rebollo-Neira, Effective high compression of ECG signals at low level distortion, Scientific Reports 9 (2019) article No. 4564. doi:10.1038/s41598-019-40350-x.

[33] L. Rebollo-Neira, Z. Xu, Sparse signal representation by adaptive non-uniform B-spline dictionaries on a compact interval, Signal Processing 90 (2010) 2308–2313. doi: https://doi.org/10.1016/j.sigpro.2010.02.004

[34] C. Tan, L. Zhang, H. Wu, A novel Blaschke unwinding adaptive Fourier decomposition based signal compression algorithm with application on ECG signals, IEEE Journal of Biomedical and Health Informatics 23 (2019) 672–682. doi:10.1109/JBHI.2018.2817192.