Optimal Resolution of Superimposed Action Potentials

Kevin C. McGill

***Abstract*—This paper presents a practical algorithm for resolving superimposed action potentials encountered during the decomposition of electromyographic signals. The problem is posed as an optimization problem: to align a set of templates with a given waveform to minimize the euclidean distance between them. The algorithm uses a recursive approach to search all possible discrete-time alignments, starting with the most likely ones and stopping once it can be verified that the optimal alignment has been found. Each candidate solution is aligned to finer-than-sampling-interval resolution using interpolation and continuous-time optimization. Both the cases in which the identities of the involved templates are known and not known are considered. Simulations are presented to show that the proposed algorithm is very accurate even for complex superpositions involving three or more similarly shaped templates, destructive interference, and added noise.**

***Index Terms*—Alignment, decomposition, electromyography, superposition, template matching.**

I. Introduction

THE electromyographic (EMG) signal recorded from a contracting muscle is made up of trains of discrete discharges known as motor-unit action potentials (MUAPs). Each MUAP is generated by a distinct group of muscle fibers, referred to collectively as a motor unit. MUAPs from different motor units tend to have distinct shapes which remain fairly constant from discharge to discharge. The MUAPs can, therefore, be identified and tracked using pattern recognition techniques, a process known as decomposition. The resulting information can be used to diagnose neuromuscular disorders, assess muscle function, and investigate motorneuron physiology.

One widely used method for identifying MUAP discharges is template matching [7], [9], [11]–[13], [17]. In this method, templates are formed for each frequently recurring discharge and then new discharges are classified according to the templates they match most closely. A difficulty arises, however, when two or more discharges overlap in time to produce a superposition that does not match any of the templates. Resolving superpositions, that is, determining the identities of the involved MUAPs and their precise times of occurrence, has been a major stumbling block in EMG decomposition.

The resolution problem can be stated as an optimization problem—namely, that of determining the templates and occurrence times that give the best fit to the superposition waveform. In theory, the solution can be found by searching all possible alignments of all possible combinations of templates [10], [16]. However, this approach is too computationally expensive for practical use.

Most proposed methods for resolving superpositions use heuristic approaches to reduce the search space in order to identify the optimal alignment quickly. One simple strategy is sequential recognition, in which the single most-likely template is identified first and then subtracted, or "peeled off," from the superposition waveform to reveal the next template and so on [2], [4], [5], [12]. Other methods consider more than one possible template alignment [3], [6], [11] or use neural networks [1]. However, these heuristic methods are not guaranteed to find the optimal solution and they may not perform well in cases involving destructive interference.

A given alignment can also be optimized using continuous-time optimization methods [3], [14]. Such methods involve interpolation and so have the added advantage of eliminating the time-quantization errors encountered in discrete-time methods. However, since the cost function usually has many local minima, a global search strategy is still necessary in order to ensure finding the global minimum.

This paper presents a practical algorithm for finding the global solution to the resolution problem. The algorithm uses a composite discrete-time and continuous-time approach. A discrete-time method is used to search the entire space of all possible discrete-time alignments. Each candidate alignment is then optimized to finer-than-sampling-interval resolution using interpolation. The algorithm is organized efficiently to try the most likely alignments first and to stop once it can be determined that the globally optimal solution has been found.

II. Statement of the Problem

A. Notation

The following notation will be used. Sets will be designated by upper-case letters. Finite sets will be represented using curly brackets and their elements will be indexed starting at zero or one as indicated in the text. R will denote the set of real numbers and R_N will denote the real interval $[0, N)$. Z_N will denote the set of integers $\{0, 1, \ldots, N-1\}$.

Sampled waveforms will be written in vector notation and designated by bold-face, lower-case letters. The elements of a waveform will be indexed starting at zero. Thus, $\mathbf{x} \in R^N$ is the vector $\mathbf{x} = [x_0, \ldots, x_{N-1}]^\dagger$ where each $x_i \in R$ and $\dagger$ indicates transpose. Waveform indexes falling outside of Z_N are assumed to wrap around, i.e.,

$$x_k \equiv x_{\mathrm{mod}_N k} \tag{1}$$

Manuscript received March 27, 2001; revised February 3, 2002.This work was supported by the Medical and Rehabilitation Research and Development Services of the U.S. Department of Veterans Affairs.

The author is with the Rehabilitation Research and Development Center, VA Palo Alto Health Care System, 3801 Miranda Ave., Palo Alto, CA 94304 USA (e-mail: mcgill@roses.stanford.edu).

Publisher Item Identifier S 0018-9294(02)05771-3.

The inner product of two vectors will be denoted $\langle \mathbf{x}, \mathbf{y} \rangle \equiv \sum_k x_k y_k$ and the norm of a vector will be denoted $||\mathbf{x}||^2 \equiv \langle \mathbf{x}, \mathbf{x} \rangle = \sum_k x_k^2$.

The interpolation operator will be denoted by a subscript: $\mathbf{x}_t$, where $t \in R_N$ is a point in time measured in sampling intervals. The interpolation operator returns the value at time t of the continuous waveform that interpolates the sample values $x_0, \ldots, x_{N-1}$. It can be implemented using the discrete fourier transform (DFT), as described in the Appendix. Note that $\mathbf{x}_t$ is a continuous scalar function of t. For an integer argument, the interpolation operator simply picks one element of the vector: $\mathbf{x}_k = x_k$.

The circular time-shift operator will be denoted by a superscript

$$\mathbf{x}^t \equiv [\mathbf{x}_{-t}, \mathbf{x}_{1-t}, \ldots, \mathbf{x}_{N-1-t}]^\dagger \tag{2}$$

where $t \in R$ and the indexes are understood to be taken $\bmod_N$. Note that $\mathbf{x}^t$ is a vector. For an integer argument, the time-shift operator simply reorders the elements of the vector: $\mathbf{x}^k = [x_{-k}, \ldots, x_{N-1-k}]^\dagger$.

B. The Known-Identities Problem

Let us consider first the case in which the identities of the templates involved in the superposition are known *a priori*. This is an important problem in practice because MUAPs discharge at fairly regular intervals and the identities of the templates can often be inferred from the identities of the other nearby discharges [4], [10], [12]. This case will also provide a framework for the case in which the identities are not known.

Let $\mathbf{x} \in R^N$ be a given superposition of n known templates plus noise

$$\mathbf{x} = \sum_{i=1}^{n} \mathbf{s}_i^{t_i^*} + \mathbf{n} \tag{3}$$

where $\mathbf{s}_i \in R^N$ is the ith template, $t_i^* \in R_N$ is its time of occurrence in units of sampling intervals and $\mathbf{n} \in R^N$ is the noise. Both the superposition waveform and the templates are assumed to have been sampled above their Nyquist rate, but not necessarily to have been oversampled. They are also assumed to be suitably zero padded to avoid wrap-around difficulties associated with circular time shifts. Note that the times of occurrence are not restricted to be integers. Given $\mathbf{x}$ and $S = \{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$, the problem is to estimate the occurrence times.

The estimate we will consider is the one that minimizes the squared euclidean distance, or "alignment error," between $\mathbf{x}$ and the reconstructed waveform. Specifically, we will attempt to find the set of occurrence times $T_C \in R_N^n$ such that

$$e(T_C) = \min_{T \in R_N^n} e(T) \tag{4}$$

where $T = \{t_1, \ldots, t_n\}$ is any set of occurrence times and where

$$e(T) \equiv \left\| \mathbf{x} - \sum_{i=1}^{n} \mathbf{s}_i^{t_i} \right\|^2 \tag{5}$$

is the alignment error. Note that the estimates T_C are not restricted to integer values. We will refer to the problem of determining T_C as the "continuous-time, known-identities" (CK) problem.

We will also be interested in finding the optimal discrete-time alignment, that is, the estimate $T_D \in Z_N^n$ such that

$$e(T_D) = \min_{T \in Z_N^n} e(T). \tag{6}$$

We will refer to this as the "discrete-time, known-identities" (DK) problem.

C. The Unknown-Identities Problem

Let us also consider the case in which the identities of the templates involved in the superposition are not known. In this case, the given waveform is the superposition of zero or more templates from a set of n known templates

$$\mathbf{x} = \sum_{i=1}^{n} w_i^* \mathbf{s}_i^{t_i^*} + \mathbf{n} \tag{7}$$

where $w_i^* \in Z_2 = \{0, 1\}$ indicates whether the ith template is involved or not. The problem in this case is to estimate both T and $W = \{w_1, \ldots, w_n\} \in Z_2^n$ to minimize the alignment error

$$e(T, W) \equiv \left\| \mathbf{x} - \sum_{i=1}^{n} w_i \mathbf{s}_i^{t_i} \right\|^2 \tag{8}$$

We will be interested both in the "continuous-time, unknown-identities" (CU) problem and the "discrete-time, unknown-identities" (DU) problem.

D. The Discrete- and Continuous-Time Approaches

The discrete-time problem is essentially a combinatorial one: to find the best alignment out of a large, but finite, set of possibilities. The challenge is to organize the search in an efficient manner so that the best alignment can be recognized quickly.

One drawback of working in discrete time is time-quantization error [14], [18]. Since the templates can only be aligned to the nearest sampling interval, considerable alignment error can remain even at the optimal discrete-time alignment. Moreover, in some situations the optimal discrete-time solution may not be close to the optimal continuous-time solution.

Time-quantization errors can be reduced in two ways: by oversampling and by interpolation. Oversampling has the drawback that it greatly increases the discrete search space. Interpolation, on the other hand, can be performed efficiently using the DFT [14] and this is the approach we will consider.

Using interpolation, the alignment error $e(T)$ can be expressed as a continous function of the continous occurrence times $t_1, \ldots, t_n$. Multidimensional continuous-time optimization techniques can then be used to minimize this function to finer-than-sampling-interval resolution. The difficulty with this approach is that it can only be used to find a local minimum in the vicinity of a given starting point. Finding the global minimum still requires some sort of global search.

The complementary advantages of the discrete- and continuous-time approaches suggest the composite approach presented in this paper. A global discrete-time search will be used to locate the neighborhood of the optimal solution and interpolation will then be used to locate it more precisely.

III. The Discrete-Time, Known-Identities Problem

This section presents an algorithm for finding the best discrete-time alignment when the template identities are known. This algorithm will also provide the basic framework for the unknown-identities and continuous-time algorithms.

Our strategy will be to try the most likely alignments first and to stop once it can be determined that the globally optimal solution has been found. This involves the following three main steps.

1) Choose a likely trial alignment of one or two templates.
2) Find the optimal alignment of all the other templates given this trial alignment.
3) Determine whether any other alignment could do any better.

As we will see, this strategy provides an efficient way to organize the algorithm, since the optimization problem in Step 2) can be solved by recursion.

The best strategy for choosing a trial alignment in Step 1) depends on the nature of the superposition. In cases involving constructive interference, a good trial alignment can be found by choosing a template that has a high cross correlation with the superposition waveform. However, in cases involving destructive interference, the superposition waveform may have less energy than the templates and may not be well correlated with any of them. In this case it would be better to look for trial offsets between pairs of templates to cancel some of their energy.

Both of these situations can be handled in an elegant way by posing the problem in a slightly more general form. Specifically, the problem can be restated as that of finding the alignment of $n+1$ waveforms (the superposition waveform and the negatives of the templates) whose sum (the energy of which equals the alignment error e) has the lowest energy. A trial alignment can be found by choosing a pair waveforms and an offset between them that reduces the energy of the sum. This procedure will automatically choose good trial alignments whether the superposition involves constructive or destructive interference.

Determining whether the globally optimal solution has been found in Step 3) can be accomplished by computing a lower bound on the alignment error over all the alignments that have not yet been tried. If no other alignment could do better than the best one found so far, then it must be optimal and the algorithm can stop. Otherwise, it is necessary to go back to Step 1) and try again.

A. The General Form

The general form of the alignment problem can be stated as follows: Given a set of $n+1$ known signals $Y = \{\mathbf{y}_0, \ldots, \mathbf{y}_n\} \in R^{N\times(n+1)}$, find the set of $n+1$ occurrence times $D = \{d_0, \ldots, d_n\} \in Z_N^{n+1}$ which minimizes the alignment error

$$e(D;Y) = \left\| \sum_{i=0}^{n} \mathbf{y}_i^{d_i} \right\|^2. \tag{9}$$

Note that we here explicitly indicate the dependence of e on the signal set Y. The DK problem can be converted to this general form by letting

$$\mathbf{y}_i = \begin{cases} \mathbf{x}, & i = 0 \\ -\mathbf{s}_i, & i = 1, \ldots, n \end{cases}. \tag{10}$$

Although d_0 equals zero in the DK problem, it will be convenient to allow the algorithm to manipulate d_0 in the same way as the other occurrence times. The solution to the DK problem can then be obtained from the solution of the general problem as follows:

$$t_i = d_i - d_0, \quad i = 1, \ldots . n. \tag{11}$$

Note that the general problem does not have a unique solution, since $e(\{d_0, \ldots, d_n\}; Y) = e(\{d_0 + a, \ldots, d_n + a\}; Y)$ for any $a \in Z_N$. However, all of these solutions map to a single solution of the DK problem.

A key fact that will be used in the algorithm is that the alignment error can be expressed in terms of cross-correlation values

$$e(D;Y) = \sum_{i,j=0}^{n} c_{ij\,d_j - d_i} \tag{12}$$

where c_{ijk} is the cross correlation between $\mathbf{y}_i$ and $\mathbf{y}_j$ at lag k

$$c_{ijk} = \langle \mathbf{y}_i, \mathbf{y}_j^k \rangle \tag{13}$$

and where the final index of c is understood to be taken $\bmod_N$. Equation (12) can be derived by expanding (9) and noting that $\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{x}^t, \mathbf{y}^t \rangle$ for any $t \in R$. The array of cross-correlation values $C = \{c_{ijk}; i, j \in Z_n, k \in Z_N\}$ only needs to be computed once at the beginning of the algorithm. This can be done efficiently using the fast fourier transform as described in the Appendix. Once these values have been computed, they provide a convenient way for computing the alignment error. They will also be useful for choosing trial alignments and estimating the error bound. Note that c_{ii0} gives the energy of signal $\mathbf{y}_i$.

B. Choosing the Trial Alignment

The first step in the algorithm strategy is to choose a trial alignment of two of the signals. Specifically, this involves choosing the triple $\{p, q, r\}$, where $p \in Z_{n+1}$ and $q \in Z_{n+1}$ are the indexes of the two waveforms and $r \in Z_N$ is the offset between them.

In order to avoid trying alignments more than once, we will keep track of which alignments have already been tried in the three-dimensional binary array $L \in Z_2^{n+1} \times Z_2^{n+1} \times Z_2^N$. Its entries will initially be set to one and when an alignment $\{p, q, r\}$ is tried, element l_{pqr} will be set to zero.

To see which alignments are good candidates for reducing the alignment error, let us rewrite (12) as the sum of the signal energies plus the interaction terms

$$e(D;Y) = \sum_i c_{ii\,0} + \sum_{i \neq j} c_{ij\,d_j - d_i}. \tag{14}$$

The sum of the signal energies is a fixed positive value that can only be reduced by including interaction terms with large negative values. One way to do this is to pick p to be the signal with the largest energy and q to be the signal that can cancel the

greatest amount of that energy. This can be formalized in the following function.

```
Function {p, q, r} = pick_dk (C, L):
 find p such that c_pp0 = max_i c_ii0;
 find q ≠ p and r such that
  c_pqr = min_{j,k} c_pjk / l_pjk.
```

The division by l_{pjk} is a shorthand notation for indicating that the minimization is restricted to alignments that have not already been tried. We will assume that expressions such as c_{pjk}/l_{pjk} evaluate to $+\infty$ if $l_{pjk} = 0$, so that these terms are effectively excluded from the minimization. Here, this prevents retrying alignments that have already been tried.

C. Solving the Constrained Problem

The second step in the strategy is to determine the optimal alignment of all the other signals given the constraint that $d_q - d_p = r$. This is equivalent to aligning the composite signal $\mathbf{y}_p + \mathbf{y}_q^r$ and the n other signals. Thus, the contstrained problem has one lower dimension than the original problem and it can, therefore, be solved by recursive application of the general algorithm.

Let us refer to the new set of signals as $\tilde{Y}$

$$\tilde{\mathbf{y}}_i = \begin{cases} \mathbf{y}_{i\lceil q} & i \neq p\lfloor q \\ \mathbf{y}_p + \mathbf{y}_q^r & i = p\lfloor q \end{cases} \quad i = 0, \ldots, n-1. \tag{15}$$

In this set, the composite signal replaces $\mathbf{y}_p$, $\mathbf{y}_q$ is eliminated and the signals are renumbered consecutively. The renumbering is accomplished by the $\lceil$ and $\lfloor$ operators, which are defined as follows: $a\lceil b \equiv a$ if $a < b$ and $a+1$ otherwise and $a\lfloor b \equiv a$ if $a < b$ and $a-1$ otherwise.

The new problem is to find $\tilde{D} = \left\{\tilde{d}_0, \ldots, \tilde{d}_{n-1}\right\}$ to minimize $e(\tilde{D}; \tilde{Y})$. The solution of the constrained problem is then given by

$$d_i = \begin{cases} \tilde{d}_{i\lfloor q} & i \neq q \\ \tilde{d}_{p\lfloor q} + r & i = q \end{cases} \quad i = 0, \ldots, n. \tag{16}$$

The cross-correlation values for the new signals will be needed for the recursion. Only those values that involve $\mathbf{y}_p$ or $\mathbf{y}_q$ are changed and these can be calculated using the formula

$$\left\langle \mathbf{y}_p + \mathbf{y}_q^r, \mathbf{y}_j^k \right\rangle = c_{pjk} + c_{qj\,k+r}. \tag{17}$$

The entire array of cross-correlation values $\tilde{C}$ for the new signals $\tilde{\mathbf{y}}$ is, thus, given by

$$\tilde{\mathbf{c}}_{ij} = \begin{cases} \mathbf{c}_{i\lceil q\,j\lceil q}, & i \neq p\lfloor q, j \neq p\lfloor q \\ \mathbf{c}_{pp} + \mathbf{c}_{qq} + \mathbf{c}_{qp}^r + \mathbf{c}_{pq}^{-r}, & i = p\lfloor q, j = p\lfloor q \\ \mathbf{c}_{p\,j\lceil q} + \mathbf{c}_{q\,j\lceil q}^r, & i = p\lfloor q, j \neq p\lfloor q \\ \mathbf{c}_{i\lceil q\,p} + \mathbf{c}_{i\lceil q\,q}^{-r}, & i \neq p\lfloor q, j = p\lfloor q \end{cases} \tag{18}$$

for $i, j = 0, \ldots, n-1$, where $\mathbf{c}_{ij} = [c_{ij\,0}, \ldots, c_{ij\,N-1}]^\dagger$.

Note also that if a particular trial alignment has already been tried in the original problem, then there is no point considering it in the constrained problem. The marker array $\tilde{L}$ for the new set of signals can, therefore, be computed as follows:

$$\tilde{\mathbf{l}}_{ij} = \begin{cases} \mathbf{l}_{i\lceil q\,j\lceil q}, & i \neq p\lfloor q, j \neq p\lfloor q \\ \mathbf{l}_{pp} \cdot \mathbf{l}_{qq}, & i = p\lfloor q, j = p\lfloor q \\ \mathbf{l}_{p\,j\lceil q} \cdot \mathbf{l}_{q\,j\lceil q}^r, & i = p\lfloor q, j \neq p\lfloor q \\ \mathbf{l}_{i\lceil q\,p} \cdot \mathbf{l}_{i\lceil q\,q}^{-r}, & i \neq p\lfloor q, j = p\lfloor q \end{cases} \tag{19}$$

for $i, j = 0, \ldots, n-1$, where $\mathbf{l}_{ij} = [l_{ij\,0}, \ldots, l_{ij\,N-1}]^\dagger$ and $\cdot$ denotes element-by-element multiplication.

D. The Lower Bound on the Alignment Error

The third step in the strategy involves calculating the lower bound of the alignment error over all the alignments that have not yet been tried. Let us denote the lowest error among all the untried alignments as e_L, where

$$e_L \equiv \min_{D \in Z_N^{n+1}} \sum_{i,j} \frac{c_{ijd_j - d_i}}{l_{ijd_j - d_i}}. \tag{20}$$

The quotient c/l is again here assumed to evaluate to $+\infty$ if $l = 0$ so that the minimization is effectively limited to those combinations of $d_0, \ldots, d_n$ for which $l_{ij\,d_j - d_i} = 1$.

To be effective, a lower bound on e_L should be tight, but also quick to compute. One simple lower bound follows from (14):

$$e_L \geq \sum_i c_{ii\,0} + \sum_{i \neq j} \min_k \frac{c_{ijk}}{l_{ijk}} \equiv b_1. \tag{21}$$

Two additional bounds are presented in the Appendix. One is a true bound like b_1, which guarantees that the algorithm will continue iterating until reaching the globally optimal solution. The other is an approximate bound. It is often tighter than the true bounds and so can speed convergence, but it sometimes overestimates e_L and so does not always guarantee finding the global solution.

The final lower bound is given by the following function.

```
Function b = lbound_dk (C, L):
 compute b_1, b_2 and b_3 from (21), (41)
  and (45);
 for the true bound: let b = max(b_1, b_2);
 for the approximate bound: let
   b = max(b_1, b_2, b_3).
```

E. The DK Algorithm

The complete algorithm for solving the DK problem can now be stated as follows.

```
Function {T, e} = align_dk(x, S):
 determine n and N from the dimensions
  of S;
 compute Y from x and S using (10);
 compute C from Y using (35);
 let l_ijk = 1 for i, j ∈ Z_{n+1}, k ∈ Z_N;
 compute {D, e} = solve_dk(C, L, n, ∞);
 compute T from D using (11).
```

```
Function {D, e} = solve_dk(C, L, n, e'):
 if n > 0:
  let D = { }, e = e';
  while lbound_dk (C, L) < e:
   let {p, q, r} = pick_dk (C, L);
   compute C̃ and L̃ using (18) and (19);
   compute {D̃, ẽ} = solve_dk(C̃, L̃, n − 1, e);
   if ẽ < e:
    compute D from D̃ using (16);
    let e = ẽ;
   let l_pqr = l_pq−r = 0;
 else if n = 0:
  let D = {0}, e = c_000.
```

The function $align\_dk$ translates between the original variables and those needed for the general function $solve\_dk$. It also computes the cross-correlation values and initializes the marker array L.

The function $solve\_dk$ solves the general problem. It tries different combinations of $\{p, q, r\}$, choosing them using the function $pick\_dk$ and solving the constrained optimization problem by calling itself recursively. The array L keeps track of which alignments are tried. e is initially set to e', which contains the smallest alignment error encountered so far by the calling program. Thereafter it keeps track of the smallest error encountered during the current call. D keeps track of the corresponding occurrence times.

The algorithm continues trying different $\{p, q, r\}$ combinations until the lower bound calculated by the function $lbound\_dk$ becomes greater than e. At this point, D and e will either contain the optimal solution of the general problem, or, if there is no solution with an alignment error less than e', D will contain a null value and e will still contain e'. These values are returned to the calling program.

At the deepest recursion level, C corresponds to a single signal which equals the sum of the original signals at their trial alignments. The alignment error e is given by the signal energy c_{000}.

IV. The Discrete-Time, Unknown-Identities Problem

The DU problem can be solved using the same recursive approach as the DK problem. In this case though, it will be necessary to consider trial template removals as well as trial alignments.

The DU problem can be restated in the more general form as follows: Given the set of known signals $Y \in R^{N\times(n+1)}$, find the set of occurrence times $D \in Z_N^{n+1}$ and the set of indicators $W \in Z_2^{n+1}$ to minimize

$$e(D, W; Y) = \left\| \sum_{i=0}^{n} w_i \mathbf{y}_i^{d_i} \right\|^2 \tag{22}$$

where the relationships between $\mathbf{y}$, $\mathbf{x}$, $\mathbf{s}$, d and t are as in (10) and (11) and $w_0 \equiv 1$.

A. Choosing the Trial Alignment or Removal

Let us keep track of the removals that have already been tried in the l_{qq0} elements of the marker array L. These elements will initially be set to one and when removal of signal q is tried, l_{qq0} will be set to zero. Element l_{000} will initially be set to zero to prevent removal of the original superposition waveform.

One strategy for deciding whether to try an alignment or a removal will be to check whether a removal would reduce the lower bound on the alignment error. This leads to the following function.

```
Function {p, q, r} = pick_du(C, L):
 determine n from the dimensions of C;
 let b = lbound_dk(C, L);
 for q = 0 to n:
  let b̃_q = lbound_dk(C̃_q, L̃_q)/l_qq0;
 if min_i b̃_i < b (trial removal):
  find q such that b̃_q = min_i b̃_i;
  let p = q, r = 0;
 else (trial alignment):
  let {p, q, r} = pick_dk(C, L).
```

Here, $\tilde{C}_q$ and $\tilde{L}_q$ are formed from C and L by removing the entries corresponding to the qth signal. This function chooses a trial removal of template q (signaled by returning the same value in p and q) if that removal has not been tried already and it would reduce the lower bound on the alignment error. Otherwise, it chooses a trial alignment using the same strategy as in the DK problem.

B. Solving the Constrained Problem

Like trial alignments, trial removals also reduce the dimensionality of the problem by one, so that recursion can be employed in both cases. Let us again refer to the reduced set of signals as $\tilde{Y} \in R^{N\times n}$. The constrained problem is to find $\tilde{D}$ and $\tilde{W}$ to minimize $e(\tilde{D}, \tilde{W}; \tilde{Y})$.

In the case of a trial alignment, the signals $\tilde{Y}$ are the same as those in the DK problem (15) and so are the formulas for the cross-correlation values $\tilde{C}$ (18) and markers $\tilde{L}$ (19). The formula for determining D from $\tilde{D}$ is also the same (16) and a similar formula applies for W

$$w_i = \begin{cases} \tilde{w}_{i\lfloor q} & i \neq q \\ \tilde{w}_{p\lfloor q} & i = q \end{cases} \quad i = 0, \ldots, n. \tag{23}$$

In the case of a trial removal, the new signals are obtained by removing signal q

$$\tilde{\mathbf{y}}_i = \mathbf{y}_{i\lceil q}, \quad i = 0, \ldots, n-1. \tag{24}$$

The formulas for updating D and W are given by

$$d_i = \begin{cases} \tilde{d}_{i\lfloor q} & i \neq q \\ 0 & i = q \end{cases} \quad i = 0, \ldots, n \tag{25}$$

$$w_i = \begin{cases} \tilde{w}_{i\lfloor q} & i \neq q \\ 0 & i = q \end{cases} \quad i = 0, \ldots, n. \tag{26}$$

The formulas for $\tilde{C}$ and $\tilde{L}$ are

$$\tilde{\mathbf{c}}_{ij} = \mathbf{c}_{i\lceil q\, j\lceil q} \quad i,j = 0,\ldots,n-1 \tag{27}$$

$$\tilde{\mathbf{l}}_{ij} = \mathbf{l}_{i\lceil q\, j\lceil q} \quad i,j = 0,\ldots,n-1. \tag{28}$$

C. The Lower Bound on the Alignment Error

For the DU problem, the lower bound must apply over all the yet untried removals as well as all the yet untried alignments. One way to do this is to compute the function *lbound_dk* for all the possible combinations of the signals Y, excluding removals of those for which removal has already been tried. This gives the following function.

```
Function b = lbound_du(C, L):
 determine n from the dimensions of C;
 let I = {i; such that l_ii0 = 1};
 let Θ be the set of all subsets of I;
 for i = 1 to the number of elements in Θ:
  let b̃_i = lbound_dk(C̃_θi, L̃_θi);
 let b = min_i b̃_i.
```

Here, $\tilde{C}_\theta$ and $\tilde{L}_\theta$ are obtained from C and L by only keeping those entries that correspond to the signals whose indexes are included in the set θ. For example, $\tilde{C}_\theta = \{c_{ijk}; i,j \in \theta, k \in Z_N\}$.

D. The DU Algorithm

The complete algorithm for solving the DU problem is

```
Function {T, W, e} = align_du (x, S):
 compute n, N, Y, C and L as in align_dk;
 let l_000 = 0;
 compute {D, W, e} = solve_du(C, L, n, ∞);
 compute T from D using (11).

Function {D, W, e} = solve_du(C, L, n, e'):
 if n > 0:
  let D = { }, e = e';
  while lbound_du (C, L) < e:
   let {p, q, r} = pick_du (C, L);
   compute C̃ and L̃ using (18)-(19) or
    (27)-(28);
   let {D̃, W̃, ẽ} = solve_du (C̃, L̃, n − 1, e);
   if ẽ < e:
    compute D from D̃ using (16) or (25);
    compute W from W̃ using (23) or (26);
    let e = ẽ;
   let l_pqr = l_pq−r = 0;
 else if n = 0:
  let D = 0, W = 1, e = c_000.
```

This is very similar to the DK algorithm. The function *solve_du* translates the original variables and initializes the cross-correlation and marker arrays. The function *solve_du* solves the general problem recursively. It tries removals or alignments chosen by *pick_du* until the lower bound becomes greater than the lowest found alignment error. The appropriate formula for $\tilde{C}$, $\tilde{L}$, D, and W must be used depending on whether a removal or an alignment is being tried. Trial removals are indicated by $\{p,q,r\}$ sets in which $p = q$.

V. The Continuous-Time, Known-Identities Problem

This section considers the continuous-time problem. We will concentrate on the known-identities (CK) case. The extension to the unknown-identities (CU) case is straightforward.

Our approach will be to use the main structure of the DK algorithm to search the space of all possible discrete-time alignments. When a likely candidate alignment is found, a continuous-time optimization procedure will be used to "fine-tune" the alignment to finer-than-sampling-interval resolution.

This approach assumes that the optimal continuous-time alignment will be reachable from at least one discrete-time alignment—that is, that there will be at least one discrete-time alignment from which the optimization procedure will converge to the optimal continuous-time alignment. This will usually be the case, due to the limited degree to which signals sampled at their Nyquist rate can vary between sampling points. Computer simulations suggest that situations in which this is not the case are likely to be very rare in practice.

A. Continuous-Time Optimization

The alignment error can be written as a continuous function of the occurrence times using the interpolated cross-correlation vectors as follows:

$$e(D) = \sum_{i,j=0}^{n} \mathbf{c}_{ij\, d_j - d_i} \tag{29}$$

where now $D \in R_N^{n+1}$. This is a well-behaved function which can be minimized using a modified Newton's method [3], [8], [14]. This method can be stated as follows.

```
Function {e, D} = optimize (D', C):
 let d = [d'_1 − d'_0, ..., d'_n − d'_0]^†;
 do:
  compute ā as in (49);
  let d = d + ā;
  until ||ā|| < 0.05/n;
 let D = {0, d_1, ..., d_n};
 compute e from D and C using (29).
```

Here, $D' \in R_N^n$ is the starting point for the optimization. The step $\bar{\mathbf{a}} \in R^n$ is computed from the gradient and Hessian of e, as described in the Appendix.

This function will be used at the deepest recursion level of the CK algorithm to optimize promising trial alignments. At that level, trial alignments have been chosen for n of the $n+1$ original signals. The occurrence times of the signals are not directly available and must be calculated from the $\{p,q,r\}$ sets at the higher recursion levels. Let us denote the values at level k by $\{p_k, q_k, r_k\}$, where $k = 1$ indicates the initial level. Then the occurrence times of the original $n+1$ signals can be computed as follows.

```
Function D = recover (n, P, Q, R):
 let D_{n+1} = 0;
 for k = n, n-1, ..., 1:
  compute D_k from D_{k+1} as in (16)
   using p_k, q_k and r_k;
 let D = D_1.
```

B. The Lower Bound on the Alignment Error

In the CK problem, the lower bound used to verify the optimality of a trial solution must apply to the continuous alignment error (29). Let us denote the lowest error in the vicinity of the untried alignments as e_{LC}, where

$$e_{LC} = \min_{\substack{D \in Z_N^{n+1} \\ \Delta \in [-1/2, 1/2]^{n+1}}} \frac{c_{ij\ d_j+\delta_j-d_i-\delta_i}}{l_{ij\ d_j-d_i}}. \tag{30}$$

In general, we can say that

$$e_{LC} \geq e_L - b_C \tag{31}$$

where e_L is the discrete-time lower bound (20) and b_C is the largest amount by which the alignment error can vary within a $\pm 1/2$ sampling-interval neighborhood. A method for estimating b_C is presented in the Appendix. This leads to the following function.

```
Function b = lbound_ck (C, L):
 global b_C;
 let b = lbound_dk (C, L) - b_C.
```

This function assumes that the correction factor b_C has already been calculated and is available as a global variable.

C. The CK Algorithm

The CK algorithm can be stated as follows.

```
Function {T, e} = align_ck (x, S):
 global n̄, C̄, D*, b_C;
 compute n̄, N, Y, C̄ and L as in align_dk;
 compute b_C as in (48);
 compute {D, e} = solve_ck (C̄, L, n̄, ∞);
 compute T from D* using (11).

Function {D, e} = solve_ck (C, L, n, e'):
 global n̄, C̄, D̄, P, Q, R;
 if n > 0:
  try trial alignments as in solve_dk,
   using lbound_ck and solve_ck and
   saving p, q, r in p_{n̄-n+1}, q_{n̄-n+1}, r_{n̄-n+1};
 else if n = 0:
  let D = { }, e = e';
  let D̄ = recover (n̄, P, Q, R);
  compute {D̄, ē} = optimize (D̄, C̄);
  if ē < e:
   let D = {0}, D* = D̄, e = ē.
```

Fig. 1. (top) Four of the MUAP spikes used in the simulations. (bottom) Four sample superpositions of these templates, showing a range from destructive (left) to constructive (right) interference.

The function *align_ck* translates and initializes the variables for the general problem. It also computes the lower-bound correction term b_C and saves it as a global variable. It also saves the original values of n and C (which are called $\bar{n}$ and $\bar{C}$). It calls *solve_ck* to solve the general problem. Upon return, it retrieves the optimal continuous-valued occurrence times from the global variable D^*.

The function *solve_ck* is similar to *solve_dk*, except that e keeps track of the smallest continuous-time alignment error and the optimality check uses the continuous-time lower-bounding function *lbound_ck*. At the deepest recursion level, the algorithm recovers the discrete occurrence times of the original templates from the $\{p, q, r\}$ values at the higher recursion levels and uses them as the starting-point for a continuous-time optimization. The best continuous-valued occurrence times found so far are stored in the global variable D^*. The discrete-valued occurrence times are saved in D for use in the discrete-time search.

The algorithm for the CU problem can be obtained by modifying the DU algorithm in a similar way.

VI. Simulations

Simulations were performed to illustrate certain aspects of the algorithms. Four experimentally recorded MUAP spikes with similar amplitudes and shapes were used as templates (Fig. 1). The spikes were averaged from an EMG signal recorded using a monopolar needle electrode during a moderate voluntary contraction of brachial biceps. The EMG signal was high-pass filtered at 1 kHz and sampled at 10 kHz. The templates were 6.4-ms long ($N = 64$), aligned by their downward-pointing peaks.

Superpositions were simulated using (3) or (7) for different numbers of templates (n). The template occurrence times t_i^* were chosen randomly from the interval [−10, 10] sampling intervals, so that the template peaks occurred within ±2 ms of one another. The noise process $\mathbf{n}$ was white Gaussian noise with a standard deviation equal to 0.05 times the mean peak-to-peak template amplitude. Because of the similarity of the templates and the proximity of the occurrence times, the resulting superpositions are typical of difficult superpositions encountered in practice. Some examples are shown in Fig. 1.

TABLE I
SIMULATION RESULTS FOR THE KNOWN-IDENTITIES PROBLEM. IDENTIFICATION RATES ($id, \%$), NUMBERS OF CALLS TO $solve$ (c_S) AND NUMBERS OF CALLS TO $optimize$ (c_O)

	Number of templates (n)								
	2			3			4		
alg.	id	c_S	c_O	id	c_S	c_O	id	c_S	c_O
dkt1	91	3.0		47	4		20	5	
dkt4	98	6.1		91	21		72	74	
dka	99	4.7		96	27		88	160	
dkt	99	8.0		96	51		93	395	
cka	100	4.8	1.2	100	26	2.4	99	165	5.1
ckt	100	8.3	1.3	100	53	2.7	100	417	5.9
exh	100	4e3		100	3e5		100	2e7	

Algorithm performance was measured in terms of identification rate and computation time. For the known-identities problem, a correct identification was counted if all the estimated occurrence times were within one sampling interval of the correct times, i.e., if

$$\max_i |t_i - t_i^*| \leq 1. \tag{32}$$

For the unknown-identities problem, correct identification further required that $w_i = w_i^*$ for all i. Computation time was measured by counting the number of recursive calls to *solve* and the number of calls to *optimize*.

A. Simulations With Known Identities

The first set of simulations involved the known-identities problem. One hundred superpositions were simulated using (3) for values of n from two to four. The $align\_dk$ and $align\_ck$ algorithms were tested using the approximate (dka, cka) and true (dkt, ckt) lower bounds. Modified versions of $solve\_dk$ were also tested in which the number of iterations (i.e., the number of $\{p, q, r\}$ sets evaluated per call) was limited to one or four ($dkt1, dkt4$). The results are shown in Table I. The final row (exh) shows the number of trial alignments that would be needed for an exhaustive search (N^n).

The $dkt1$ algorithm is representative of the peeling-off approach: it considers only one single trial alignment chosen by the maximum cross-correlation criterion. Although very fast, it was not reliable for superpositions involving more than two templates. Identification performance was improved when four trial alignments per recursion level were considered ($dkt4$), but this number was not always sufficient to find the optimal alignment.

The dkt algorithm continued trying alignments until the optimality of the solution had been verified. It found the correct alignment (to within ± 0.5 sampling interval) most of the time, but not always, due to time quantization error. The true lower bound used by dkt tended to be conservative and essentially equal identification performance was achieved with fewer computations using the approximate bound (dka).

TABLE II
SIMULATION RESULTS FOR THE UNKNOWN-IDENTITIES PROBLEM. IDENTIFICATION RATES ($id, \%$), NUMBERS OF CALLS TO $solve$ (c_S) AND NUMBERS OF CALLS TO $optimize$ (c_O). TOTAL NUMBER OF TEMPLATES (n) = 4

	Number of involved templates (n_I)								
	1			2			3		
alg.	id	c_S	c_O	id	c_S	c_O	id	c_S	c_O
dua	98	348		98	387		81	294	
dut	98	676		97	695		84	564	
cua	100	485	39	100	462	15	99	328	8.0
cut	100	878	44	100	792	16	99	611	8.4

The continuous-time algorithms (ckt and cka) almost always found the correct alignment (to within ± 0.1 samling intervals), with computation counts similar to their discrete-time counterparts. Note that only a relatively small number of continuous-time optimizations actually needed to be performed. The correct solution can also always be found by exhaustive search (exh), but this is computationally prohibitive.

B. Simulations With Unknown Identities

The second set of simulations involved the unknown-identities problem. One hundred superpositions were simulated using (7) for different numbers of involved templates $n_I = \sum w_i$ out of a total of $n = 4$ templates. The $align\_du$ and $align\_cu$ algorithms were tested using the approximate (dua, cua) and true (dut, cut) lower bounds. The results are shown in Table II.

All the algorithms required more computations than in the $n = 4$ known-identities simulation, regardless of the number of templates involved in the superposition, since all combinations of templates needed to be considered. In fact, the number of computations increased as the number of involved templates decreased. This is because superpositions involving only a few templates were difficult to distinguish from destructive superpositions involving more templates. The continuous-time algorithm had almost perfect identification rates, even using the approximate lower bound.

VII. DISCUSSION

This paper presents a practical algorithm for resolving superimposed waveforms. The algorithm determines the optimal alignment in the sense that it gives the best fit between the sum of the templates and the superposition waveform. It, therefore, provides greater accuracy than many previously proposed methods which do not guarantee the optimal solution.

The algorithm searches the space of all possible alignments using an efficient, recursive approach which finds and verifies the optimal solution quickly, even in cases involving destructive

interference. Time-quantization errors are avoided by using interpolation to fine tune the optimal alignment to finer-than-sampling-interval accuracy.

Separate versions of the algorithm are presented for the cases in which the identities of the templates involved in the superposition are known and not known *a priori*. The unknown-identities problem requires greater computational effort due to the need to consider all possible template combinations. Moreover, the computation count grows exponentially with the number of templates that need to be considered. For this reason, the use of outside information such as discharge histories to identify the templates involved in a superposition should be preferred to the indiscriminant use of the unknown-identities algorithm.

The accuracy of the algorithm depends on the signal-to-noise ratio (SNR) and the similarity or lack of similarity between the templates. Actual MUAPs can also exhibit waveform variability from discharge to discharge which was not considered here, but which will also affect identification performance.

The computation count depends on the SNR, the number of templates and the template similarity. The simulated superpositions depicted in Fig. 1 are difficult cases given the similarity of the templates, the high degree of overlap and the level of noise. Lower computation counts could be expected if the template shapes were more different from one another.

The computation count also depends to a large extent on the the strategies used for picking the trial alignments and estimating the lower bounds. The strategies presented here work fairly well for both constructive and destructive interference. However, it is likely that even better strategies could be devised.

We have been using the cka algorithm routinely as part of an interactive program for decomposing EMG signals [15]. We frequently decompose signals containing 12–16 active MUAPs, in which superpositions involving three or more MUAPs are common. We have found the cka algorithm to perform well across a wide variety of superpositions involving similarly shaped templates, complexly shaped (polyphasic) templates and templates with a wide range of amplitudes. The algorithm is implemented in Matlab on a Macintosh G3 computer and typically requires less than one second to resolve a superposition involving five templates.

APPENDIX

A. Calculation of the Interpolation Operator

The interpolation operator can be implemented using the DFT as follows:

$$\mathbf{x}_t \equiv \frac{1}{N} \sum_{k=-N/2+1}^{N/2-1} e^{j2\pi kt/N} \xi_{\mathrm{mod}_N k} \tag{33}$$

where $\boldsymbol{\xi} = [\xi_0, \ldots, \xi_{N-1}]^\dagger \equiv \mathrm{FFT}(\mathbf{x})$ is the fast fourier transform of $\mathbf{x}$. Note that the frequency index k in (33) ranges from $-N/2+1$ to $N/2+1$ rather than from zero to $N-1$ to prevent aliasing.

The derivatives of $\mathbf{x}_t$ needed for computing the gradient and Hessian of the alignment error in the CK problem can be computed as follows:

$$\frac{d^i \mathbf{x}_t}{dt^i} = \frac{1}{N} \sum_{k=-N/2+1}^{N/2-1} \left(\frac{j2\pi k}{N}\right)^i e^{j2\pi kt/N} \xi_{\mathrm{mod}_N k}. \tag{34}$$

B. Calculation of Cross-Correlation Values

The cross-correlation values $\mathbf{c}_{ij} = [c_{ij\,0}, \ldots, c_{ij\,N-1}]^\dagger$ for two signals $\mathbf{x}_i$ and $\mathbf{x}_j$ can be computed efficiently using the fast fourier transform as follows:

$$\mathbf{c}_{ij} = \mathrm{FFT}^{-1}(\boldsymbol{\xi}_i \cdot \boldsymbol{\xi}_j^*) \tag{35}$$

where $\boldsymbol{\xi}_i = \mathrm{FFT}(\mathbf{x}_i)$, $\cdot$ denotes element-wise multiplication and $*$ denotes complex conjugation.

C. Additional Lower Bounds for the DK Problem

All the bounds we will consider involve the minimum values of c_{ijk} over the untried alignments. Therefore, let us define

$$m_{ij} = \begin{cases} c_{ii0} & i = j \\ \min_k c_{ijk}/l_{ijk} & i \neq j \end{cases} \tag{36}$$

for $i, j \in Z_{n+1}$. Note that the bound b_1 in (21) can be written as

$$b_1 = \sum_{i,j=0}^{n} m_{ij}. \tag{37}$$

A second lower bound for the DK problem can be established as follows. Let $\mathbf{y} = \mathbf{y}_i^{d_i}$ and $\mathbf{u} = \sum_{j \neq i} \mathbf{y}_j^{d_j}$ for some i, so that $e(D;Y) = \|\mathbf{y}+\mathbf{u}\|^2$. Since $\langle \mathbf{y}, \mathbf{u}\rangle \leq \|\mathbf{y}\|\,\|\mathbf{u}\|$, it follows that:

$$\begin{aligned} e(D;Y) \geq & \|\mathbf{y}\|^2 + 2\langle \mathbf{y}, \mathbf{u}\rangle + \frac{\langle \mathbf{y}, \mathbf{u}\rangle^2}{\|\mathbf{y}\|^2} \\ = & \frac{\langle \mathbf{y},\, \mathbf{y}+\mathbf{u}\rangle^2}{\|\mathbf{y}\|^2.} \end{aligned} \tag{38}$$

Now, $\langle \mathbf{y},\ \mathbf{y}+\mathbf{u}\rangle = \sum_j c_{ijd_j-d_i} \geq \sum_j m_{ij}$. Since $\sum_j m_{ij}$ can be positive or negative, it follows that:

$$\langle \mathbf{y},\ \mathbf{y}+\mathbf{u}\rangle^2 \geq \max\left(0, \sum_{j=0}^{n} m_{ij}\right)^2 \tag{39}$$

and, therefore, that

$$e_L \geq \frac{\max\left(0, \sum_{j=0}^{n} m_{ij}\right)^2}{\|\mathbf{y}_i\|^2} \tag{40}$$

for any i. This leads to the following bound:

$$e_L \geq \max_i \frac{\max\left(0, \sum_{j=0}^{n} m_{ij}\right)^2}{\|\mathbf{y}_i\|^2} \equiv b_2. \tag{41}$$

An approximate lower bound can also be established. Let $\mathbf{y}$ and $\mathbf{u}$ be defined as above. From the model (3) we have

$\sum_j \mathbf{y}_j^{d_j^*} = \mathbf{n}$, where the d_j^* are the true alignments and n is noise. Therefore

$$\mathbf{y} = \mathbf{n}^{d_i - d_i^*} - \sum_{j \neq i} \mathbf{y}_j^{d_j^* - d_i^* + d_i} \equiv \mathbf{n}' - \mathbf{u}^*. \tag{42}$$

Note that $\mathbf{n}'$ and $\mathbf{u}^*$ are time-shifted versions of the noise and the sum of the signals $\mathbf{y}_j$, $j \neq i$, at their true alignments. Thus

$$e(D;Y) = \langle \mathbf{y}, \mathbf{y} + \mathbf{u} \rangle + \langle \mathbf{u}, \mathbf{u} - \mathbf{u}^* \rangle + \langle \mathbf{u}, \mathbf{n}' \rangle. \tag{43}$$

The second term in this expression is likely to be greater than zero, since for most sets D, $||\mathbf{u}||^2 > \langle \mathbf{u}, \mathbf{u}^* \rangle$. The third term equals the correlation of the sum of templates with noise. If the signals and the noise are both zero mean, then the expected value of this term will be zero. Therefore, we can say

$$e(D;Y) \gtrsim \langle \mathbf{y}, \mathbf{y} + \mathbf{u} \rangle \tag{44}$$

where by $\gtrsim$ we mean that this inequality is expected to hold for many sets D. We, therefore, have

$$e_L \gtrsim \max_i \sum_{j=0}^{n} m_{ij} \equiv b_3. \tag{45}$$

During execution of *align_dk*, all three bounds tend to be quite loose at first, since the matrix of m_{ij} values is not necessarily a cross-correlation matrix. The bounds become tighter as the alignments with the largest negative c_{ijk} values are tried and removed from the m_{ij} calculations. b_1 is often tighter than b_2 for small values of n (two or three), but b_2 is often tighter for larger values. b_3 is usually tighter than both, but it sometimes overestimates e_L and so cannot guarantee that the optimal alignment will be found.

D. The Continuous-Time Lower-Bound Correction

The term b_C corrects the discrete-time lower bound on the alignment error by taking into account the amount by which the error can vary between the discrete sampling points.

Let D_C be a local minimum of $e(D)$ and let D_D be the discrete alignment with the smallest alignment error within ± 1 sampling intervals of D_C. We are looking for an upper bound on $e(D_D) - e(D_C)$. Since D_C is a local minimum, $e(D)$ can be expanded about D_C to obtain

$$e(D_D) = e(D_C) + (\mathbf{d}_D - \mathbf{d}_C)^\dagger \mathbf{H} (\mathbf{d}_D - \mathbf{d}_C) + \cdots \tag{46}$$

where $\mathbf{d}_C = [d_{C1}, \ldots, d_{Cn}]^\dagger$, $\mathbf{d}_D = [d_{D1}, \ldots, d_{Dn}]^\dagger$ and $\mathbf{H} = [\partial^2 e / \partial d_i \partial d_j]_{i,j=1}^n$ is the Hessian matrix. In general, we expect that $|d_{Ci} - d_{Di}| < 1/2$ and, therefore, that

$$e(D_D) - e(D_C) \leq \left(\frac{1}{2}\right)^2 \text{trace } \mathbf{H}. \tag{47}$$

Experience shows that this formula tends to be much too conservative. We have found that the following formula gives satisfactory results:

$$b_C = \left(\frac{1}{2}\right)^2 \max_{i \neq j} \max_k \; c_{ij\,k-1} - 2c_{ij\,k} + c_{ij\,k+1}. \tag{48}$$

That is, b_C can be set to 1/4 times the maximum value of the second-order difference of the cross-correlation between any of the original signals.

E. Computing the Optimization Step

The step $\bar{\mathbf{a}} \in R^n$ for Newton's method is given by

$$\bar{\mathbf{a}} = \frac{\mathbf{a}}{\max(||\mathbf{a}||, 1)} \tag{49}$$

with $\mathbf{a} \in R^n$ being the solution of

$$\bar{\mathbf{H}} \mathbf{a} = \mathbf{g} \tag{50}$$

where $\mathbf{g} \in R^n$ is the gradient

$$g_i \equiv \frac{\partial e}{\partial d_i} = -2 \sum_{j=0}^{n} \frac{d\mathbf{c}_{ij\,t}}{dt}\bigg|_{d_j - d_i} \qquad i = 1, \ldots, n \tag{51}$$

and $\bar{\mathbf{H}} \in R^n \times R^n$ is the modified Hessian matrix, which is computed from the actual Hessian $\mathbf{H}$ using spectral decomposition or modified Cholesky factorization [8]. The actual Hessian is given by

$$h_{ij} \equiv \frac{\partial^2 e}{\partial d_i \partial d_j} = \begin{cases} -2 \dfrac{d^2 \mathbf{c}_{ij\,t}}{dt^2}\Big|_{d_j - d_i} & i \neq j \\ 2 \displaystyle\sum_{\substack{k=0 \\ k \neq i}}^{n} \dfrac{d^2 \mathbf{c}_{ik\,t}}{dt^2}\Big|_{d_k - d_i} & i = j \end{cases} \tag{52}$$

for $i, j = 1, \ldots, n$.

F. Practical Considerations

Certain redundancies were ignored in this paper to simplify the didactic presentation. The algorithms can be made more efficient by taking them into account. For one thing, the cross-correlation values are symmetric in that $c_{ijk} = c_{ji\,-k}$. Therefore, only half of them need to be computed and stored. Also, the c_{iik} values are not needed for $k \neq 0$. Finally, since the signals are real, the formulas for the interpolation operator and its derivatives can be rewritten to involve only the the DFT coefficients corresponding to the nonnegative frequencies.

REFERENCES

[1] R. Chandra and L. M. Optican, "Detection, classification and superposition resolution of action potentials in multiunit single-channel recordings by an on-line real-time neural network," *IEEE Trans. Biomed. Eng.*, vol. 44, pp. 403–412, May 1997.

[2] C. I. Christodoulou and C. S. Pattichis, "Unsupervised pattern recognition for the classification of EMG signals," *IEEE Trans. Biomed. Eng.*, vol. 46, pp. 169–178, Feb. 1999.

[3] R. J. P. De Figueiredo and A. Gerber, "Separation of superimposed signals by a cross-correlation method," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 1084–1089, 1983.

[4] H. Etawi and D. Stashuk, "Resolving superimposed motor unit action potentials," *Med. Biol. Eng. Comput.*, vol. 34, pp. 33–40.

[5] J. Fang, G. C. Agarwal, and B. T. Shahani, "Decomposition of multiunit electromyographic signals," *IEEE Trans. Biomed. Eng.*, vol. 46, pp. 685–697, June 1999.

[6] J. Fang, J. Ben-Arie, Z. Wang, G. Agarwal, and B. T. Shahani, "Separation of superimposed EMG potentials using expansion matching technique," in *Proc. 19th Intl. Conf. IEEE Eng. Med. Biol. Soc.*, 1997, pp. 1585–1588.

[7] A. Gerber, R. M. Studer, R. J. P. de Figueiredo, and G. S. Moschytz, "A new framework and computer program for quantitative EMG signal analysis," *IEEE Trans. Biomed. Eng.*, vol. BME-31, pp. 285–295, 1984.

[8] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*. London, U.K.: Academic, 1981.

[9] P. Guiheneuc, J. Calamel, C. Doncarli, D. Gitton, and C. Michel, "Automatic detection and pattern recognition of single motor unit potentials in needle EMG," in *Computer Aided Electromyography*, J. E. Desmedt, Ed. Basel, Switzerland: Karger, 1983.

[10] R. Gut and G. S. Moschytz, "High-precision EMG signal decomposition using communication techniques," *IEEE Trans. Signal Processing*, vol. 48, pp. 2487–2494, Sept. 2000.
[11] W. F. Haas and M. Meyer, "An automatic EMG decomposition system for routine clinical examination and clinical research—ARTMUP," in *Computer-Aided Electromyography and Expert Systems*, J. E. Desmedt, Ed. Amsterdam, The Netherlands: Elsevier Science, 1989.
[12] R. S. Lefever and C. J. De Luca, "A procedure for decomposing the myoelectric signal into its constituent action potentials: I. technique, theory and implementation," *IEEE Trans. Biomed. Eng.*, vol. BME-29, pp. 149–157, 1982.
[13] K. C. McGill, K. L. Cummins, and L. J. Dorfman, "Automatic decomposition of the clinical electromyogram," *IEEE Trans. Biomed. Eng.*, vol. BME-32, pp. 470–477, 1985.
[14] K. C. McGill and L. J. Dorfman, "High-resolution alignment of sampled waveforms," *IEEE Trans. Biomed. Eng.*, vol. BME-31, pp. 462–468, 1984.
[15] K. C. McGill and Z. C. Lateva, "Accurate estimation of motor-unit action potential waveforms and discharge patterns," *Clin. Neurophysiol.*, vol. 110, pp. S250–S251, 1999.
[16] V. J. Prochazka and H. H. Kornhuber, "On-line multi-unit sorting with resolution of superposition potentials," *Electroenceph. Clin. Neurophysiol.*, vol. 34, pp. 91–93, 1973.
[17] D. Stashuk and H. De Bruin, "Automatic decomposition of selective needle-detected myoelectric signals," *IEEE Trans. Biomed. Eng.*, vol. 35, pp. 1–10, Jan. 1988.
[18] B. C. Wheeler and W. J. Heetderks, "A comparison of techniques for classification of multiple neural signals," *IEEE Trans. Biomed. Eng.*, vol. BME-29, pp. 752–759, 1982.

Kevin C. McGill received the A.B., B.S., and M.S. degrees from the University of Notre Dame, Notre Dame, IN, in 1974, 1975, and 1979, respectively. He received the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 1984.

Since 1984, he has been a Biomedical Engineer with the Rehabilitation Research and Development Center, VA Palo Alto Health Care System and since 1995 a Consulting Associate Professor in the Department of Functional Restoration, Stanford University School of Medicine. His research interests include neuromuscular physiology, electromyography, and signal processing.