UNIWERSYTET
WARSZAWSKI

Uniwersytet Warszawski
*Wydział Matematyki, Informatyki i Mechaniki*

PRACA DOKTORSKA W DYSCYPLINIE INFORMATYKA

---

# Heterogeniczna implementacja matching pursuit z symulacją ciągłej przestrzeni parametrów

Autor pracy:
**Piotr Tadeusz Różański**

Promotorzy:
**prof. dr hab. Piotr Durka**
**prof. dr hab. Krzysztof Stencel**

A. D. MMXXIV

## Streszczenie

Spośród metod analizy sygnałów, w analizie sygnałów bioelektrycznych (w tym EEG) istotną rolę odgrywa algorytm matching pursuit. Pomimo licznych prac teoretycznych na temat tego algorytmu i jego wariantów, niewiele uwagi poświęcono dotychczas technicznym aspektom jego działania, takim jak konstrukcja słownika (czyli zbioru funkcji, z którego korzysta algorytm) i jego wpływ na wyniki. Co więcej, brak ogólnie dostępnej, wysoce wydajnej implementacji algorytmu (zwłaszcza jego wersji wielozmiennej) utrudnia praktyczne obliczenia na dużych zbiorach danych, co szczególnie dotyczy np. diagnostyki medycznej.

Niniejsza rozprawa doktorska składa się z czterech powiązanych tematycznie artykułów naukowych, opublikowanych w recenzowanych czasopismach o zasięgu międzynarodowym. Artykuły te skupiają się na różnych aspektach obliczeń z wykorzystaniem algorytmu matching pursuit, związanych zarówno z jego wydajnością, jak i z własnościami matematycznymi. W pracy przedstawiono optymalną konstrukcję słownika dla algorytmu matching pursuit i zbadano jej właściwości. W połączeniu z lokalną optymalizacją parametrów, pozwoliło to na uzyskanie symulacji ciągłej przestrzeni parametrów, a w konsekwencji — eliminację wpływu struktury słownika na wyniki dekompozycji. Uzyskane wyniki zostały wdrożone w stworzonej od podstaw i opublikowanej na licencji open source implementacji matching pursuit *empi*, łączącej w sobie obsługę obliczeń wielowątkowych na procesorach konwencjonalnych (CPU) i graficznych (GPU).



## Abstract

Among signal analysis methods, the matching pursuit algorithm plays a crucial role in the analysis of bioelectrical signals, including EEG. Despite numerous theoretical studies on this algorithm and its variants, little attention has been paid to the technical aspects of its operation, such as the construction of the dictionary (i.e., the set of functions used by the algorithm) and its impact on results. Moreover, the lack of a publicly available, highly efficient implementation of the algorithm (particularly its multivariate version) hampers practical computations on large datasets, as is often required in medical diagnostics.

This doctoral dissertation comprises a collection of four thematically related scientific articles published in peer-reviewed international journals. These articles focus on various aspects of computations using the matching pursuit algorithm, addressing both its performance and its mathematical properties. The optimal construction of the dictionary for the matching pursuit algorithm is presented, and its properties are examined. Combined with local parameter optimization, this approach enabled the simulation of a continuous parameter space, thereby completely eliminating the effect of the dictionary structure on the decomposition results. The obtained results were implemented in an open-source matching pursuit implementation *empi*, supporting multi-threaded computations on conventional (CPU) as well as graphics (GPU) processors.

**Oświadczenia kierujących pracą**

Oświadczam, że niniejsza praca została przygotowana pod moim kierunkiem i stwierdzam, że spełnia ona warunki do przedstawienia jej w postępowaniu o nadanie stopnia doktora nauk ścisłych i przyrodniczych w zakresie dyscypliny informatyka.

| | |
|---|---|
| Data | prof. dr hab. Piotr Durka |
| | prof. dr hab. Krzysztof Stencel |

**Oświadczenie autora pracy**

Świadom odpowiedzialności prawnej oświadczam, że niniejsza rozprawa doktorska została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem stopnia doktora w innej jednostce.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

| | |
|---|---|
| Data | Piotr Tadeusz Różański |

# Spis treści

Serdecznie dziękuję obydwu Promotorom za możliwość korzystania z bardzo szerokiej autonomii w pracy badawczej i implementacyjnej, a jednocześnie za gotowość do realnej pomocy, kiedykolwiek była potrzebna — szczególnie w gorącym okresie finalizacji pracy. Prof. Durce szczególnie dziękuję za możliwość udziału w rozlicznych projektach badawczych i badawczo-rozwojowych, związanych z szeroko pojętą analizą sygnałów biomedycznych.

Do przygotowania niniejszego dokumentu wykorzystany został system XeTeX z użyciem szablonu autorstwa Federico Maggiego, za co również (ponownie!) winien jestem autorowi wspomnianego szablonu serdeczne *Grazie!*

# Wstęp 1

## 1.1 Motywacja pracy

Bezpośrednią motywacją do podjęcia tej tematyki były wieloletnie badania prowadzone na Wydziale Fizyki Uniwersytetu Warszawskiego od lat 90. XX wieku. Prace te koncentrowały się na parametryzacji i algorytmicznej analizie sygnałów bioelektrycznych, takich jak magnetoencefalogram (MEG), elektrokardiogram (EKG) i przede wszystkim elektroencefalogram (EEG).

Elektroencefalografia (EEG) jest jedną z kluczowych metod badania aktywności mózgu, znajdującą szerokie zastosowanie zarówno w diagnostyce medycznej, jak i w badaniach naukowych. Przez wiele lat standardem w analizie EEG była wzrokowa ocena zapisów przez lekarzy i specjalistów. Choć podejście takie pozwala na identyfikację istotnych wzorców, jest ono subiektywne i mało powtarzalne (Durka 2007). Z tego powodu coraz większą uwagę zwraca się na algorytmiczną parametryzację struktur obecnych w EEG, co pozwala na bardziej obiektywną i powtarzalną analizę danych.

Od lat 90. XX wieku jedną z metod analizy danych stosowanych z powodzeniem w EEG jest algorytm matching pursuit (Mallat i Z. Zhang 1993). Matching pursuit to metoda dekompozycji sygnału, która iteracyjnie dopasowuje elementy ze zdefiniowanego słownika funkcji do analizowanych danych, minimalizując błąd rekonstrukcji na każdym etapie. Zalety tego algorytmu obejmują jego zdolność do adaptacyjnego dopasowywania się do lokalnych cech sygnału oraz efektywność w analizie sygnałów o skomplikowanej strukturze (Durka 2007).

Jednym z kluczowych wyzwań w efektywnym stosowaniu algorytmu matching pursuit jest zdefiniowanie odpowiedniego słownika, czyli zbioru funkcji, z którego wybierane będą elementy używane do dekompozycji sygnału. Wybór słownika ma decydujący wpływ na jakość rekonstrukcji sygnału oraz na zdolność algorytmu do uchwycenia istotnych cech analizowanego EEG. Kolejnym istotnym problemem jest wydajność obliczeń — analiza dużych zbiorów danych EEG, zwłaszcza w kontekście wielozmiennych (ang. *multi-variate*) sygnałów, wymaga zastosowania wysoce wydajnych implementacji algorytmów.

W obliczu tych wyzwań, celem niniejszej rozprawy było opracowanie i implementacja wydajnych i bardziej dokładnych metod analizy danych z wykorzystaniem algorytmu mat-

ching pursuit. Proponowane rozwiązania mają na celu poprawę efektywności obliczeń oraz dokładności analizy poprzez odpowiedni dobór słownika i optymalizację procesów obliczeniowych.

## 1.2 Zadania badawcze

Tematykę niniejszej pracy stanowi opracowanie i implementacja wydajnej metody obliczeń algorytmem matching pursuit, a w szczególności następujące zadania badawcze:

**Z1.** opracowanie i zbadanie własności optymalnej struktury słownika funkcji dla algorytmu matching pursuit,

**Z2.** wprowadzenie symulacji słownika ciągłego poprzez połączenie idei optymalnego słownika z lokalną optymalizacją parametrów,

**Z3.** stworzenie wysoce zoptymalizowanej implementacji algorytmu z obsługą GPU.

## 1.3 Plan niniejszej pracy

Główną część pracy stanowią cztery recenzowane artykuły naukowe dołączone w formie sekcji **A**-**D**.

Niniejszy tekst podzielony jest na trzy rozdziały, stanowiące wstęp i syntezę wprowadzającą w główną część pracy, przedstawiając cele, metody i wyniki zarówno badań teoretycznych jak i działań implementacyjnych nad algorytmem matching pursuit.

Niniejszy rozdział 1. przedstawia motywację do podjęcia badań oraz zarysowuje główne wyzwania związane z analizą sygnałów bioelektrycznych za pomocą algorytmu matching pursuit.

Rozdział 2. wprowadza podstawowe informacje na temat analizy szeregów czasowych i metod analizy sygnałów. W szczególności, przedstawia historię algorytmu matching pursuit, jego zastosowanie w analizie sygnałów EEG i istniejące wyzwania w jego stosowaniu.

Rozdział 3. skupia się na podsumowaniu wkładu autora w rozwój implementacji matching pursuit, przedstawiając uzyskane wyniki w kilku różnych, kluczowych dla implementacji algorytmu, obszarach i w nawiązaniu do zadań badawczych zdefiniowanych powyżej.

# Wiadomości wstępne 2

## 2.1 Analiza szeregów czasowych

Szeregi czasowe to sekwencje danych uporządkowanych w czasie, które umożliwiają analizę i modelowanie dynamicznych procesów. Niniejsza praca skupia się na szeregach dyskretnych próbkowanych równomiernie oraz pomija problem kwantyzacji sygnału. Zakładając te ograniczenia, szereg czasowy można zdefiniować jako funkcję z $t_s\mathbb{N}$ w $\mathbb{R}$, gdzie

$$t_s\mathbb{N} = \{0, t_s, 2t_s, 3t_s, \ldots\}, \tag{2.1}$$

$t_s$ jest okresem próbkowania sygnału, a $f_s = \frac{1}{t_s}$ jego częstością próbkowania.

Szeregi czasowe znajdują szerokie zastosowanie w różnych dziedzinach nauki, takich jak:

- ekonomia, w analizie technicznej cen akcji i innych instrumentów finansowych;
- meteorologia, w analizie i predykcji temperatury, ciśnienia, prędkości wiatru etc.;
- astronomia, w analizie zmienności obserwowanych własności ciał niebieskich, takich jak położenia, jasności, skład widmowy;
- oraz w niezliczonych innych zastosowaniach.

W szczególności, w kontekście sygnałów biomedycznych, szeregi czasowe odgrywają kluczową rolę w medycynie i diagnostyce, umożliwiając analizę sygnałów takich jak EEG (elektroencefalografia), EKG (elektrokardiografia), EMG (elektromiografia), fMRI (funkcjonalny jądrowy rezonans magnetyczny) i wielu innych. Analiza tych sygnałów pozwala z jednej strony na wykrywanie patologii i stanów chorobowych, a z drugiej strony na monitorowanie stanu zdrowia pacjentów (Niedermeyer i Silva 2005). Nie do przecenienia jest również potencjał analizy sygnałów biomedycznych w diagnostyce osób z zaburzeniami świadomości (Hannawi i in. 2015).

Tradycyjnie, analiza szeregów czasowych (zwanych dalej zamiennie sygnałami) opierała się na metodach takich jak Dyskretna Transformacja Fouriera (DFT) oraz będąca jej efektywną implementacją Szybka Transformacja Fouriera (FFT) (Cooley i Tukey 1965). Metody te pozwalają na dekompozycję sygnału na składowe harmoniczne, co jest szczególnie

przydatne w analizie sygnałów stacjonarnych. DFT i FFT mają jednak dość ograniczoną skuteczność w analizie sygnałów niestacjonarnych, ponieważ nie dostarczają informacji o zmienności czasowej składowych częstościowych.

By przezwyciężyć te ograniczenia, wprowadzono analizę falkową, która umożliwia dekompozycję sygnału na składowe o zmiennej rozdzielczości czasowo-częstościowej (Daubechies 1992). Analiza falkowa pozwala na bardziej efektywne uchwycenie lokalnych cech sygnałów niestacjonarnych, co czyni ją szczególnie przydatną w analizie sygnałów biomedycznych (Addison 2002). Transformacja falkowa jest szczególnie użyteczna w analizie sygnałów o zmieniających się częstościach, jednak również ma swoje własne ograniczenia, takie jak wybór odpowiedniej funkcji falkowej oraz problem zmiennej rozdzielczości w dziedzinie czasu i częstości.

Jako remedium na ograniczenia przywołanych powyżej metod zaproponowano algorytm matching pursuit (Mallat i Z. Zhang 1993) który pokonuje niektóre z tych ograniczeń poprzez adaptacyjne dopasowywanie elementów słownika do analizowanego sygnału. Algorytm ten umożliwia bardziej elastyczną i precyzyjną analizę szeregów czasowych, co jest szczególnie ważne w kontekście analizy skomplikowanych sygnałów biomedycznych.

## 2.2 Historia matching pursuit

Zaproponowany ponad trzydzieści lat temu (Mallat i Z. Zhang 1993) matching pursuit (MP) jest algorytmem służącym do aproksymacji sygnałów poprzez przedstawienie ich w postaci sumy prostszych funkcji zwanych atomami czas-częstość. MP jest częścią szerszej kategorii tzw. *sparse approximations*, które dążą do reprezentacji sygnałów w sposób możliwie najbardziej oszczędny.

Formalnie, algorytm na podstawie sygnału $x(t)$ i ustalonego zbioru funkcji (sygnałów) $D$ zwanego **słownikiem** (a same funkcje zwane będą dalej zamiennie **atomami**), wyznacza przybliżenie sygnału w postaci kombinacji liniowej atomów

$$x(t) \approx \sum_i c_i \, g_i(t) \,, \quad \text{gdzie } g_i \in D) \,. \tag{2.2}$$

Algorytm działa w sposób iteracyjny i można go opisać następująco:

1. **Inicjalizacja:** Rozpoczynamy od sygnału $x(t)$ i pustego zbioru atomów.

2. **Projekcja:** W każdej iteracji wybieramy atom $g(t)$ ze słownika, który maksymalizuje wartość projekcji sygnału na ten atom.

3. **Aktualizacja:** Odejmujemy od sygnału $x(t)$ jego projekcję na wybrany atom, w rezultacie otrzymując resztę, czyli sygnał rezydualny $x'(t)$.

Proces jest powtarzany iteracyjnie, w każdej iteracji biorąc jako początkowy sygnał resztę z poprzedniej iteracji $x^{(i-1)}(t)$. Iteracja powtarzana jest aż do osiągnięcia zadanej liczby iteracji lub spełnienia kryterium zatrzymania, np. gdy łączna energia reszty spadnie poniżej pewnego poziomu.

Matematycznie, dla sygnału $x$ oraz słownika atomów $D$, procedura zdefiniowana jest jako

$$\begin{aligned} x^{(0)} &= x \\ g_i &= \operatorname{argmax}_{g \in D}(g \cdot x^{(i-1)}) \\ c_i &= g_i \cdot x^{(i-1)} \\ x^{(i)} &= x^{(i-1)} - c_i \, g_i \,, \end{aligned} \tag{2.3}$$

gdzie $\cdot$ jest iloczynem skalarnym zdefiniowanym dla sygnałów dyskretnych jako

$$x \cdot y = \sum_t x(t) \, y(t) \tag{2.4}$$

gdzie sumowanie przebiega po całej dziedzinie czasu.

Choć w ogólnym przypadku słownik $D$ może składać się z dowolnych funkcji (o ile spełniają one warunek normalizacyjny $g \cdot g = 1$), już w oryginalnej pracy Mallata (1993) zaproponowane zostały funkcje (atomy) Gabora, czyli oscylujące funkcje z gaussowską obwiednią, zdefiniowane jako

$$g_{(s,f_0,t_0)}(t) = K_{(s,f_0,t_0)} \, \exp\left(-\pi \frac{(t-t_0)^2}{s^2}\right) \cos(2\pi f(t-t_0) + \varphi) \,, \tag{2.5}$$

gdzie $K_{(s,f_0,t_0)}$ jest odpowiednio dobranym czynnikiem normalizacyjnym. Sprowadza to problem konstrukcji słownika do znalezienia optymalnego próbkowania trójwymiarowej przestrzeni parametrów $(s, f_0, t_0)$, gdzie $s$ odpowiada szerokości (długości trwania w czasie) atomu, $f_0$ jego częstości oscylacji, zaś $t_0$ maksimum obwiedni. Fazę $\varphi$ można pominąć na tym etapie, gdyż dla zadanych $(s, f_0, t_0)$ i konkretnego sygnału $x$, da się już w ramach samej procedury matching pursuit wyznaczyć fazę $\varphi$ maksymalizującą iloczyn skalarny $g \cdot x$.

Nie bez związku z transformatą falkową, Mallat i Zhang (1993) zaproponowali próbkowanie diadyczne (ang. *dyadic sampling*), w którym skalę $s$ tworzą ciąg geometryczny[1] $s_j = 2^j$, zaś próbkowanie w czasie i częstości spełnia $\Delta_t \sim s$, $\Delta_f \sim \frac{1}{s}$. Jak łatwo zauważyć, tak powstały słownik będzie zawierał porównywalną liczbę funkcji dla każdej skali $s_j$. Konstrukcja ta nie precyzuje jednak, jak gęste ma być próbkowanie w czasie i częstości — będzie to jednym z problemów analizowanych w niniejszej pracy.

W ciągu lat opracowano wiele wariantów i rozszerzeń algorytmu matching pursuit, aby poprawić jego dokładność, zbieżność i wydajność w różnych zastosowaniach. Orthogonal Matching Pursuit (OMP) to rozszerzenie matching pursuit, które zapewnia ortogonalność wybieranych atomów, co poprawia zbieżność i dokładność, kosztem zwiększonego czasu obliczeń. OMP iteracyjnie wybiera atom maksymalnie skorelowany z resztą i ortogonalizuje resztę względem wszystkich wcześniej wybranych atomów. Metoda ta zwiększa wydajność algorytmu, szczególnie w środowiskach szumowych i zastosowaniach w kompresyjnym próbkowaniu (Tropp i Gilbert 2007; Chi i Calderbank 2012). Algorytm Generalized Orthogonal Matching Pursuit (gOMP) rozszerza OMP poprzez wybieranie wielu atomów w każdej iteracji zamiast jednego. Podejście to przyspiesza zbieżność i poprawia wydajność

[1] Podstawę ciągu skal można naturalnie zastąpić dowolną inną liczbą większą od 1.

odzyskiwania rzadkich sygnałów (J. Wang, Kwon i Shim 2011). Stagewise Orthogonal Matching Pursuit (StOMP) jest kolejnym wariantem, który przetwarza sygnał etapami, pozwalając na dodanie wielu współczynników na każdym etapie na podstawie mechanizmu progowania. Metoda ta jest szczególnie użyteczna dla sygnałów o wysokim stopniu rzadkości i okazała się skuteczna w różnych zastosowaniach, w tym w rekonstrukcji obrazów (Donoho i in. 2012).

Opracowano kilka odpornych wariantów OMP, aby radzić sobie z wartościami odstającymi i szumem nie-Gaussowskim. Na przykład Robust OMP (RobOMP) wykorzystuje estymatory M, aby zmniejszyć wpływ wartości odstających (Loza 2019). Ponadto Tuning-Free OMP (TF-OMP) eliminuje konieczność znajomości z góry rzadkości sygnału lub wariancji szumu, co czyni go bardziej praktycznym dla rzeczywistych zastosowań (Kallummil i Kalyani 2017). Różne algorytmy OMP zostały szeroko analizowane pod kątem ich gwarancji wydajności w różnych warunkach. Przykładowo, badano wydajność w obecności szumu, wykazując, że OMP może niezawodnie odzyskać rzadkie sygnały przy spełnieniu określonych warunków dotyczących macierzy pomiarowej (Cai i L. Wang 2011). Ponadto wykazano skuteczność algorytmu w zastosowaniach takich jak rekonstrukcja obrazów i przetwarzanie sygnałów (Bian i L. Zhang 2021).

Warto zauważyć, że niezależnie od zastosowanego wariantu, kluczowym elementem wpływającym na jakość dekompozycji jest (a) struktura słownika i (b) efektywność implementacji, czyli te dwie kwestie, którym poświęcona została główna część niniejszej pracy.

Algorytm matching pursuit i jego warianty znalazły szerokie zastosowanie w różnych dziedzinach nauki i inżynierii. Przykładem jest użycie matching pursuit do rozkładu sygnałów na sinusoidy tłumione, co jest przydatne w identyfikacji systemów, estymacji spektralnej i modelowaniu sygnałów (Goodwin 1997). W dziedzinie telekomunikacji warianty algorytmu używane są w kompresji danych audio i wideo dzięki zdolności do efektywnej reprezentacji sygnałów (Chen, Donoho i Saunders 2001), zaś w analizie obrazów do kompresji, rekonstrukcji oraz odszumiania (Rubinstein, Bruckstein i Elad 2010). W komunikacji bezprzewodowej algorytm matching pursuit jest wykorzystywany do estymacji kanałów, estymacji kierunku przybycia (DOA) oraz detekcji wielodostępowej (MUD). Inne algorytmy z tej rodziny, takie jak Flexible Tree-Based OMP (FTB-OMP), oferują niską złożoność obliczeniową i są skuteczne w problemach detekcji i estymacji w systemach bezprzewodowych (Karabulut 2006).

Matching pursuit został również zastosowany do rozszerzenia fourierowskiej metody *split-operator* do reprezentacji stanów koherentnych w symulacjach procesów kwantowych. Algorytm ten pozwala na dokładne i efektywne symulacje głębokiego tunelowania i długoczasowej dynamiki (Wu i Batista 2003).

## 2.3 Matching pursuit w analizie EEG

Algorytm matching pursuit znalazł szerokie zastosowanie w analizie sygnałów biomedycznych, szczególnie w analizie sygnałów EEG (elektroencefalogramów). Jego zdolność do dokładnej dekompozycji sygnałów w dziedzinie czasu i częstości uczyniła go użytecznym narzędziem w wielu badaniach klinicznych i neurofizjologicznych.

Jednym z pierwszych zastosowań algorytmu w analizie EEG było badanie przejściowych zdarzeń w sygnałach EEG, takich jak wrzeciona snu. Algorytm pozwalał na precyzyjne lokalizowanie tych zdarzeń w płaszczyźnie czas-częstość i śledzenie ich ewolucji w czasie

(Durka i Blinowska 1995; Shu-ren 2003; Loza i Príncipe 2016). Zastosowanie matching pursuit do analizy nocnego EEG pozwoliło na ilościowe określenie zmian mocy i częstości występujących w zapisach struktur w zależności od głębokości snu (Malinowska i in. 2007), również dla osób z zaburzeniami świadomości (Zieleniewska i in. 2019).

Algorytm matching pursuit został również zastosowany do analizy aktywności napadów padaczkowych, szczególnie w analizie rozkładu czasowo-częstościowego sygnałów EEG podczas napadów. Pozwoliło to na identyfikację charakterystycznych wzorców aktywności w różnych fazach napadu (Franaszczuk i in. 1998). Algorytm został również wykorzystany do analizy różnic między normalnym EEG a EEG podczas takich napadów, wykazując istotne różnice w rozkładzie czasowo-częstościowym (Xiao i Zheng 2012) oraz pozwalając na ich automatyczną klasyfikację (Giannakaki i in. 2019).

Kolejnym zastosowaniem algorytmu jest analiza reakcji EEG na bodźce wibracyjne, co pozwoliło na dokładne śledzenie odpowiedzi w korze somatosensorycznej na stymulację dotykową (Durka, Kelly i Blinowska 1996; Żygierewicz i in. 1998). Również w analizie zapisów magnetoencefalograficznych (MEG) wykazano wysoką skuteczność algorytmu matching pursuit w detekcji komponentu M100 (Jörn i in. 2011).

W kontekście EEG na przywołanie zasługuje również wariant Consensus Matching Pursuit (CMP), opracowany w celu analizy sygnałów EEG charakteryzujących się zmiennością między próbami (Bénar i in. 2009). Innym sposobem uwzględnienia zmienności pomiędzy kanałami i próbami jest wielozmienny wariant algorytmu MP ze zmienną fazą (Durka, Malinowska i in. 2015).

Mimo dość dobrego ugruntowania matching pursuit w analizie sygnałów biomedycznych, z większości powyższych prac przebija brak jednolitej, efektywnej implementacji algorytmu. Autorzy korzystają zazwyczaj ze swoich własnych implementacji, często napisanych w językach interpretowanych (Python, R, Matlab) bez szczególnych optymalizacji. Najpopularniejsza implementacja matching pursuit, czyli MPTK (Krstulovic i Gribonval 2006), nie obsługuje obliczeń równoległych, co w kontekście nowoczesnych stacji obliczeniowych bardzo ogranicza ich możliwości.

Podobny problem dotyczy szczegółów technicznych obliczeń, w tym struktury słownika stosowanego do dekompozycji, który w większości amatorskich lub pół-profesjonalnych implementacji musi być zadany ręcznie. Wiąże się z tym konieczność kompromisu pomiędzy dokładnością a czasem obliczeń, jak również ryzyko, że dla ustalonego rozmiaru słownika jego struktura będzie nieoptymalna i może istotnie zniekształcać wyniki obliczeń (Durka, Ircha i Blinowska 2001). Zaproponowanie wysoce wydajnej implementacji z optymalną strukturą słownika niewpływającą na wyniki obliczeń jest w związku z tym nadrzędnym celem niniejszej pracy.

# Opis wkładu pracy doktorskiej 3

Główną część pracy stanowią załączone jako dodatki **A**-**D** cztery recenzowane artykuły naukowe, z których trzy (**B**-**D**) są publikacjami jednoautorskimi, zaś w pierwszej z nich (**A**) wkład poszczególnych autorów jest szczegółowo opisany w sekcji *Authors' contributions*.

Artykuły poruszają wiele różnych wątków związanych zarówno z samym algorytmem matching pursuit, jak i z konkretnymi zastosowaniami (np. biomedycznymi). Główne wątki, wokół których skupiają się najważniejsze osiągnięcia niniejszej pracy, zostaną opisane poniżej.

## 3.1 Optymalna konstrukcja słownika

Jedną z podstawowych trudności w praktycznym zastosowaniu algorytmu matching pursuit jest zaproponowanie konstrukcji słownika, czyli optymalnego próbkowania funkcji w przestrzeni parametrów $(s, f_0, t_0)$, gdzie $s$ jest szerokością atomu (w czasie), $f_0$ częstością oscylacji, zaś $t_0$ położeniem w czasie. Konstrukcję taką można przeprowadzić w oparciu o metrykę odległości wyprowadzoną z iloczynu skalarnego sygnałów tj.

$$d(g_0, g_1) = \frac{1}{\sqrt{2}}||g_0 - g_1|| = \sqrt{1 - g_0 \cdot g_1} \tag{3.1}$$

wymagając, aby odległość pomiędzy sąsiednimi atomami (w sensie powyższej metryki) we wszystkich trzech wymiarach $(s, f_0, t_0)$ była taka sama, równa arbitralnemu parametrowi $\epsilon$, stanowiącemu miarę rzadkości słownika. Takiej konstrukcji, zwanej słownikiem optymalnym (ang. *optimal dictionary*), której zarys został pierwotnie zaprezentowany w pracy magisterskiej Marka Barwińskiego (2004), poświęcona jest znacząca część artykułu **A**.

Dla atomów Gabora, czyli oscylujących funkcji z gaussowską obwiednią (2.5), z zapro-

ponowanej konstrukcji słownika wynikają następujące formuły próbkowania:

$$\begin{aligned} s_j &= s_0\, a^j \\ a &= \exp\left(\operatorname{arcosh}\frac{1}{(1-\epsilon^2)^2}\right) = \frac{1+\epsilon\sqrt{(2-\epsilon^2)(\epsilon^4-2\epsilon^2+2)}}{(1-\epsilon^2)^2} \\ \Delta_f &= \frac{1}{s}\sqrt{-\frac{2}{\pi}(1-\epsilon^2)} \\ \Delta_t &= s\sqrt{-\frac{2}{\pi}(1-\epsilon^2)}\,. \end{aligned} \tag{3.2}$$

W szczególności, do uzyskania powyższego wyniku wymagane jest zastosowanie przybliżenia dużych częstości $sf_0 \gg 1$, co pozwala na eliminację wpływu fazy $\varphi$ na formuły iloczynu skalarnego. Tylko w takim przypadku możliwe jest uzyskanie próbkowania zgodnego z ideą słownika diadycznego Mallata. Uzyskany w ten sposób słownik ma również inną bardzo praktyczną zaletę. Przedstawiając iloczyn skalarny sygnału $x$ i wybranego atomu Gabora $g$ jako

$$x \cdot g = K_{(s,f_0,t_0)} \sum_t x(t) \exp\left(-\pi\frac{(t-t_0)^2}{s^2}\right) \cos(2\pi f(t-t_0)+\varphi)\ , \tag{3.3}$$

można zauważyć, że wyrażenie to można wyznaczyć przy pomocy dyskretnej transformaty cosinusowej, a więc z użyciem szybkiej transformacji Fouriera (FFT). Co więcej, przy pomocy pojedynczego obliczenia FFT można otrzymać wartości $x \cdot g$ od razu dla wszystkich możliwych częstości $f_0$, o ile próbkowanie słownika w częstości ($\Delta_f$) pokrywa się z krokiem wyznaczonego spektrogramu. Z tego względu zabieg ten wymaga zazwyczaj dodatkowego wypełnienia zerami (ang. *zero-padding*) analizowanego sygnału, ale co do zasady obliczenia iloczynu skalarnego sprowadzają się do wyznaczenia odpowiednio dostosowanego spektrogramu, czyli krótkoczasowej transformaty Fouriera (ang. *short-time Fourier transform*).

Szczegółowa analiza własności numerycznych implementacji matching pursuit MP5 (Kuś, Różański i Durka 2013), w której została zaimplementowana konstrukcja słownika z artykułu **A** wykazała problemy numeryczne w zbieżności dekompozycji, które na drodze analizy udało się powiązać z występowaniem atomów o wysokich częstościach, zbliżonych do częstości Nyquista ($\approx \frac{1}{2} f_s$). W artykule **B** zaproponowano efektywny, choć przybliżony, sposób radzenia sobie z normalizacją atomów o wysokich częstościach, które, z uwagi na dyskretyzację w czasie, wymagają odpowiedniej adaptacji czynnika normalizacyjnego $K_{(s,f_0,t_0)}$.

Artykuł **B** zawiera również opis szeregu istotnych kwestii obliczeniowych. Po pierwsze, poprzez wprowadzenie zespolonych atomów Gabora wraz z ich odmienną normalizacją, proponuje konkretny sposób powiązania równania (3.3) z wynikami obliczeń FFT. Po drugie, wskazuje na uproszczony sposób wyznaczania optymalnej fazy $\varphi$ maksymalizującej iloczyn skalarny. Po trzecie, przedstawia wybrane szczegóły implementacyjne (np. zastosowanie kolejki priorytetowej) w pierwszej, dość wczesnej (0.4.1) wersji implementacji *empi*.

Rozszerzeniu konstrukcji słownika na funkcje inne niż atomy Gabora poświęcony jest artykuł **C**. Przedstawia on, w systematyczny sposób, konstrukcję słownika dla dowolnej rodziny funkcji oscylujących, ograniczonych arbitralną (a nie tylko gaussowską) obwiednią, przedstawiając również wyniki dla szeregu konkretnych obwiedni (np. trójkątnej czy Lorentza). Artykuł zawiera również studium przypadku dla zaproponowanych „meta-wykładniczych" obwiedni, wskazując, że możliwe jest skonstruowanie nie-gaussowskiej obwiedni, dla której

dla tej samej wartości parametru $\epsilon$ wymagana gęstość słownika będzie istotnie mniejsza, niż dla słownika złożonego z atomów Gabora.

Efekt ten zbadany został w artykule **C** nie tylko teoretycznie, ale także na konkretnym przykładzie dekompozycji matching pursuit sygnałów pochodzących z nagrań audio. Analizując liczbę iteracji wymaganą do uzyskania założonego poziomu dokładności, na podstawie trzech niezależnych przykładów udało się wskazać, że wykorzystanie meta-wykładniczych obwiedni istotnie zmniejsza wymaganą liczbę iteracji algorytmu, a co więcej, dostosowanie konstrukcji słownika do tej konkretnej obwiedni pozwala na dalsze poprawienie uzyskanego wyniku.

Techniczne aspekty przeprowadzenia obliczeń w oparciu o FFT dla funkcji o arbitralnej obwiedni stanowią jedno z zagadnień poruszonych w artykule **D**. Artykuł ten podejmuje ponownie problem normalizacji i optymalnej fazy, tym razem opierając się na ścisłych formułach z artykułu **C** zakłada od razu dyskretną postać obwiedni, co pozwala na uzyskanie ścisłej (z maszynową dokładnością) normalizacji funkcji i uniknięcie efektów dyskretyzacji w czasie. Artykuł podaje również ścisłe formuły na powiązanie wyników FFT z poszukiwanymi wartościami iloczynów skalarnych pomiędzy sygnałem a atomami ze słownika, jak również optymalną fazą atomu. Wszystkie podane formuły zostały zaimplementowane w finalnej wersji implementacji *empi* i szczegółowo przetestowane pod względem numerycznym. Wraz z artykułem **C** stanowi to rozwiązanie zadania badawczego **Z1**.

## 3.2 Symulacja ciągłej przestrzeni parametrów

Już we wprowadzającym matching pursuit artykule (Mallat i Z. Zhang 1993) pojawia się rozróżnienie na słownik idealny $D_\infty$ (tamże oznaczany jako $\mathcal{D}$) zawierający potencjalnie nieskończoną liczbę atomów i na skończony słownik $D$ (tamże oznaczany jako $\mathcal{D}_\alpha$) stosowany do praktycznych obliczeń z uwagi na numeryczny charakter algorytmu. Autorzy wprowadzają przy tym parametr $\alpha$ stanowiący miarę jakości słownika, taki, że dla dowolnego sygnału $x$ zachodzi[1]

$$\sup_{g \in D} x \cdot g \geqslant \alpha \sup_{g_\infty \in D_\infty} x \cdot g_\infty \,. \tag{3.4}$$

Innymi słowy — im parametr $\alpha$ jest bliższy jedności, z tym większą dokładnością dekompozycja z wykorzystaniem słownika $D$ zbliża się do dekompozycji, jaką otrzymalibyśmy z wykorzystaniem słownika pokrywającego w sposób ciągły całą przestrzeń parametrów.

W tej samej publikacji zarysowany jest sposób na wyjście poza ograniczenia skończonego słownika. Autorzy proponują wykonanie w każdej iteracji algorytmu, po znalezieniu najlepiej pasującego elementu $g \in D$, lokalnej optymalizacji w przestrzeni parametrów w celu znalezienia elementu $\hat{g}$, dla którego występuje lokalne maksimum iloczynu skalarnego $x \cdot g$. Naturalnie, wtedy

$$x \cdot \hat{g} \geqslant x \cdot g \,. \tag{3.5}$$

Autorzy nie sugerują jednak (i słusznie), jakoby w ten sposób efektywnie symulowali obliczenia z wykorzystaniem słownika idealnego $D_\infty$. Należy bowiem zauważyć, że o ile otrzymane w ten sposób rozwiązanie $\hat{g}$ wskazuje w każdej iteracji maksimum lokalne iloczynu skalarnego, to niekoniecznie stanowi również maksimum globalne. Wprowadzenie

[1] Ponieważ zawsze można wykonać podstawienie $g \to -g$, wartości bezwzględne z cytowanej pracy zostały tu pominięte.

tylko lokalnej optymalizacji zniweczyłoby podstawową zasadę algorytmu, polegającą na tym, że w każdej iteracji dopasowujemy najlepiej pasujący element z zadanego słownika funkcji.

W kontekście lokalnej optymalizacji, bardziej użyteczne jest zdefiniowanie parametru $\alpha$ w sposób lokalny, jako minimalną wartość gwarantująca, że

$$\forall_{g_\infty \in D_\infty} \exists_{g \in D}\, x \cdot g \geqslant \alpha (x \cdot g_\infty)\,. \tag{3.6}$$

Warto zauważyć, że w przeciwieństwie do parametru $\alpha$ w postaci wprowadzonej w pracy Mallata (Mallat i Z. Zhang 1993), definicja (3.6) nie ma charakteru globalnego oszacowania maksymalnego błędu pojedynczej iteracji algorytmu matching pursuit; ma jedynie charakter lokalny.

Wprowadzając oznaczenie $\varepsilon(g)$ jako podzbiór elementów $D_\infty$, dla których $g$ jest najlepszym przybliżeniem tzn. jest wybierane przez kwantyfikator szczegółowy w (3.6), można zapisać

$$\alpha = \min_{g \in D} \min_{g_\infty \in \varepsilon(g)} \frac{x \cdot g}{x \cdot g_\infty}\,. \tag{3.7}$$

Oznacza to, że jeśli znany jest parametr $\alpha$ dla zadanej realizacji słownika $D$, to znając wartość $x \cdot g$ dla danego elementu słownika $g$, można znaleźć górne ograniczenie na wartości iloczynu skalarnego pomiędzy sygnałem a funkcjami (atomami) z najbliższego otoczenia $g$. W szczególności, możemy określić, ile maksymalnie może wynosić $x \cdot \hat{g}$ (gdzie $\hat{g}$ jest wynikiem lokalnej optymalizacji parametrów startującej z $g$) na podstawie wartości $x \cdot g$, jako

$$\forall_{g \in D} \forall_{\hat{g} \in \varepsilon(g)} \forall_{g_\infty \in \varepsilon(g)}\, x \cdot \hat{g} \leqslant x \cdot g_\infty \leqslant \frac{1}{\alpha}\, x \cdot g\,. \tag{3.8}$$

Zaproponowanym w artykule **D** kluczem do znajdowania globalnego maksimum jest wykonanie lokalnej optymalizacji parametrów, ale nie tylko od elementu maksymalizującego $x \cdot g$, ale również z innych elementów $g \in D$, dla których lokalna optymalizacja może dać potencjalnie lepszy wynik. Praktycznym rozwiązaniem jest iteracja po potencjalnych kandydatach w kolejności malejącego iloczynu skalarnego, tj.

**function** FindGlobalSolution
  $g \leftarrow \operatorname{argmax}_{g \in D}\, x \cdot g$
  $\hat{g}_{\text{best}} \leftarrow$ LocalOptimize$(g)$
  $G \leftarrow D \setminus \{g\}$
  **while** $G \neq \varnothing$ **do**
    $g \leftarrow \operatorname{argmax}_{g \in G}\, x \cdot g$
    **if** $\frac{1}{\alpha}(x \cdot g) < x \cdot \hat{g}_{\text{best}}$ **then**
      **break** ▷ startując z $g \in G$ nie osiągniemy wyniku lepszego niż $x \cdot \hat{g}_{\text{best}}$
    **end if**
    $\hat{g} \leftarrow$ LocalOptimize$(g)$
    **if** $x \cdot \hat{g} > x \cdot \hat{g}_{\text{best}}$ **then**
      $\hat{g}_{\text{best}} \leftarrow \hat{g}$
    **end if**
    $G \leftarrow G \setminus \{g\}$
  **end while**
  **return** $\hat{g}_{\text{best}}$
**end function**

Główną trudność stanowi zatem określenie parametru $\alpha$ dla zadanej realizacji (a więc także i konstrukcji) słownika. W postaci (3.7) wymagałoby wysoce niepraktycznego analizowania wszystkich możliwych sygnałów. Można natomiast zauważyć, że jeśli zrzutujemy dowolny sygnał $x$ na $g_\infty$, możemy go przedstawić jako

$$x = (x \cdot g_\infty) g_\infty + \xi \,, \tag{3.9}$$

gdzie $\xi$ jest składową sygnału ortogonalną do $g_\infty$ ($\xi \cdot g_\infty = 0$). Formuła (3.7) sprowadzi się zatem do

$$\alpha = \min_{g \in D} \min_{g_\infty \in \varepsilon(g)} \left( g_\infty \cdot g + \frac{\xi \cdot g}{x \cdot g_\infty} \right) . \tag{3.10}$$

Ponieważ zakładamy, że $g_\infty$ znajduje się w najbliższym otoczeniu $g$, a $\xi \cdot g_\infty = 0$, możemy zaniedbać składnik po prawej stronie, otrzymując dzięki temu przybliżeniu

$$\alpha = \min_{g \in D} \min_{g_\infty \in \varepsilon(g)} (g_\infty \cdot g) \,. \tag{3.11}$$

Zagadnienie wyznaczenia parametru $\alpha$ dla konkretnej realizacji słownika można więc sprowadzić do badania relacji pomiędzy atomami słownika $g$ a elementami przestrzeni parametrów $g_\infty$ w ich najbliższym otoczeniu.

Pierwszy krok na drodze do szacowania własności słownika został wykonany już w pracy **A**, gdzie na potrzeby optymalnej konstrukcji słownika analizowane były odległości (a więc i iloczyny skalarne) pomiędzy atomami słownika dla wszystkich trzech wymiarów ($s$, $f_0$, $t_0$). Nie jest to jeszcze jednak poszukiwany wynik, z dwóch powodów. Po pierwsze, w ramach analizy zastosowane zostało przybliżenie wysokich częstości ($sf_0 \gg 1$) jako warunek konieczny uzyskania regularnej struktury słownika. Po drugie, analizowane były jedynie zależności pomiędzy atomami sąsiadującymi ze sobą i różniącymi się dokładnie jednym parametrem, podczas gdy w równaniu (3.11) należy wziąć pod uwagę również bardziej ogólne konfiguracje.

Dokładniejsze rozwiązanie tego problemu zostało uwzględnione w artykule **D**. Z uwagi na wysoce nieliniowe efekty uniemożliwiające efektywne podejście analityczne, przeprowadzona została numeryczna analiza własności słownika optymalnego i parametru $\alpha$ dla szerokiego przedziału różnych wartości parametru rzadkości słownika $\epsilon$. Wyniki wskazują, że dla słownika optymalnego wartość parametru $\alpha$ można rozłożyć na dwa czynniki, z których jeden zależny jest tylko od $\epsilon$ (i to dość liniowo), zaś drugi jest istotny tylko dla atomów niespełniających przybliżenia wysokich częstości i zależy jedynie od iloczynu $sf_0$. W ramach artykułu zaproponowane zostały również praktyczne oszacowania na parametr $\alpha$ (a konkretnie, $\alpha^2$) użyteczne dla praktycznych obliczeń numerycznych. Stanowi to zarówno teoretyczne, jak i użytkowe rozwiązanie zadania badawczego **Z2**.

Wnioski z analizy numerycznej i wydajnościowej zawartej w artykule **D** (Fig. 8) wskazują, że dzięki symulacji słownika ciągłego jakość dekompozycji jest niezależna od parametru $\epsilon$ wybranego dla konstrukcji słownika. Możliwy jest więc wybór takiej wartości parametrów (np. $\epsilon^2 = 0.05$) dla której czas obliczeń jest minimalny, jednocześnie eliminując wpływ struktury słownika na wyniki dekompozycji.

## 3.3 Zastosowania i rys historyczny

Pierwszym krokiem w implementacji algorytmu było wdrożenie konstrukcji słownika optymalnego do rozwijanej na Wydziale Fizyki UW implementacji MP5[2]. Algorytm w tej wersji został przedstawiony w artykule **A** (Kuś, Różański i Durka 2013), a następnie wykorzystany do szeregu eksperymentów i analiz, w szczególności do analizy wrzecion snu (Durka, Malinowska i in. 2015). W obu wzmiankowanych tu publikacjach przedstawiono, poza wynikami typowo naukowymi, również rozwijane na Uniwersytecie i we współpracy z firmą BrainTech graficzne środowisko Svarog do analizy sygnałów, którego jedną z istotnych funkcji była i jest dekompozycja matching pursuit.

Z uwagi na ograniczenia istniejącej implementacji, pierwsza, dość ograniczona (w stosunku do obecnych możliwości) wersja nowej implementacji *empi* powstała w 2015 roku w ramach grantu z Funduszu Inicjatyw Dydaktycznych. Była to wersja już zrównoleglona, choć nadal niepozbawiona pewnych problemów (np. numerycznych). Już na tym etapie zrezygnowano z proponowanego przez Mallata (1993) i wykorzystywanego w implementacji MP5 schematu aktualizacji iloczynku skalarnego (sekcja *Product update formula* w **A**) zastępując go przeliczaniem iloczynów skalarnych $x \cdot g$ dla wybranych atomów na nowo. Pozwoliło to na znaczące zmniejszenie zapotrzebowania pamięciowego, jak również na uniknięcie narastających błędów numerycznych wynikających z niedokładności analitycznych wyrażeń na iloczyn skalarny atomów Gabora. Po rozwiązaniu większości problemów numerycznych (związanych również z odpowiednim traktowaniem czynnika normalizacyjnego) ta wstępna wersja implementacji (0.4.1) została zaanonsowana w ramach artykułu **B**, a w wersji 0.4.3 wykorzystana do analizy efektów obwiedni przedstawionych w artykule **C**.

W tej wersji implementacja *empi* została wykorzystana do analizy profili EEG pacjentów z zaburzeniami świadomości (Zieleniewska i in. 2019), w ramach której udało się wykazać, że algorytmiczna analiza w oparciu o algorytm *matching pursuit* może być wartościowym narzędziem diagnostycznym, nie mniej dokładnym od wizualnej analizy eksperckiej. Kluczem do tych wyników była nie tylko odpowiednia wydajność implementacji, pozwalająca na przetworzenie wielu całonocnych nagrań EEG, ale również jej wsteczna kompatybilność z poprzednikiem (MP5), dzięki której udało się zachować możliwość współpracy ze wspomnianym już środowiskiem graficznym Svarog.

W kolejnych latach kod implementacji został stopniowo przepisany w całości, wprowadzając bardziej optymalny model paralelizacji obliczeń, korzystający z funkcji współczesnych wersji języka C++ (np. *std::thread*). Dla większej stosowalności i łatwości użycia, domyślny format pliku dekompozycji został zmieniony z binarnego formatu stosowanego przez MP5 na format SQLite, lub opcjonalnie, JSON. Z analogicznego powodu zmieniony został również format wejścia — ustawienia, zamiast w tekstowych plikach konfiguracyjnych, przekazywane są obecnie bezpośrednio jako parametry linii poleceń.

Warto zauważyć, że implementacja zaczyna być również stosowana przez autorów niezwiązanych z Uniwersytetem Warszawskim. Przykładowo, z jej wykorzystaniem przeprowadzono analizę zapisów EEG i MEG osób w średnim wieku (Gula i in. 2021), wskazując na istotną zmienność w natężeniu fal alfa i beta wraz z wiekiem. Implementacja została również wspomniana w podręczniku do analizy sygnałów (Blinowska i Żygierewicz 2022) wydanym przez CRC Press.

Jednym z niedawnych zastosowań implementacji są wyniki dwu-etapowej analizy wie-

[2] https://github.com/BrainTech/matching-pursuit

lokanałowych zapisów EEG pod kątem niewykrywalnych dotąd wrzecion snu (Durka, Dovgialo i in. 2024), z wykorzystaniem symulacji słownika ciągłego. Ilość danych analizowanych w ramach tej pracy wskazuje na to, że uzyskana w ramach prac nad implementacją wydajność obliczeniowa stanowiła jeden z warunków koniecznych realizowalności projektu:

> *«(...) the implementation of the MMP algorithm (empi), which allowed us to efficiently decompose $\sim 10^3$ h of multivariate EEG into $\sim 10^7$ time-frequency atoms, from which sleep spindles were chosen in this study.»* (Durka, Dovgialo i in. 2024)

## 3.4 Wydajna implementacja z obsługą GPU

Obecna wersja implementacji *empi* opisana jest szczegółowo w artykule **D** oraz bezpośrednio w kodzie źródłowym[3]. Kluczowym pod względem wydajności elementem algorytmu jest aktualizacja iloczynów skalarnych $x \cdot g$ pomiędzy sygnałem residualnym z kolejnych iteracji a atomami słownika. Dość naturalne jest spostrzeżenie, że wystarczy aktualizować iloczyny jedynie dla tych atomów, które pokrywają się ze zmienionym w ostatniej iteracji fragmentem sygnału. Z uwagi na znaczną liczbę atomów w słowniku oraz jego regularną strukturę, obliczenia te można efektywnie podzielić pomiędzy wiele wątków CPU i GPU.

Jak zostało to już wskazane w sekcji 3.1, wyrażenie na iloczyny skalarne pomiędzy sygnałem a atomami można, pod warunkiem starannego uwzględnienia efektów normalizacji, obliczyć z pomocą okienkowanej szybkiej transformaty Fouriera. Do obliczeń na CPU została w tym celu wykorzystana implementacja FFTW (Frigo i Johnson 2005) — na czas obliczeń uruchamiana jest określona liczba wątków pobocznych odpowiedzialnych za obliczenia FFT, z którymi wątek główny komunikuje się na zasadzie producent-konsument.

Obliczenia GPU są natomiast obsługiwane przez bibliotekę CuFFT wchodzącą w skład zestawu bibliotek numerycznych CUDA (NVIDIA, Vingelmann i Fitzek 2020). O ile FFTW działa optymalnie obliczając pojedyncze transformaty po kolei, o tyle CUDA oferuje największy zysk wtedy, gdy obliczenia wykonywane są w sposób masowo równoległy. Aby sprostać temu zapotrzebowaniu, w ramach *empi* zaimplementowana została metoda obliczania krótko-czasowej okienkowanej transformaty Fouriera w oparciu o CuFFT. Jako kluczowy element kodu, pozwalający na przemnażanie poszczególnych fragmentów sygnału $x$ przez obwiednie atomów Gabora (lub innych), został wykorzystany mechanizm tzw. *callbacks*, czyli fragmentów kodu kompilowanych i uruchamianych w specyficzny sposób, umożliwiających wykonywanie dodatkowych czynności w trakcie działania procedur CuFFT.

Obok wątków CPU uruchamiane są więc także wątki obsługujące dostępne procesory graficzne i zarówno jedne, jak i drugie pobierają zlecenia obliczeniowe od wątku głównego. Z uwagi na inny charakter obliczeń i fakt, że GPU wykazuje lepszą wydajność przy analizowaniu długich fragmentów sygnału, kolejka zleceń ma charakter listy posortowanej po długości fragmentu sygnału, z której wątki CPU wybierają zlecenia poczynając od najkrótszego, zaś wątki GPU — od najdłuższego.

Zgodnie z analizą wydajnościową zawartą w artykule **D**, zastosowanie jednej lub dwu kart CUDA z obsługą podwójnej precyzji zmniejsza czas obliczeń nawet kilkukrotnie w stosunku do obliczenia na samym 16-rdzeniowym CPU. Implementacja *empi* wraz z towarzyszącym opisem i podbudową teoretyczną, stanowi rozwiązanie zadania badawczego **Z3**.

[3] https://github.com/develancer/empi

# Bibliografia


Paul S. Addison (2002). *The Illustrated Wavelet Transform Handbook: Introductory Theory and Applications in Science, Engineering, Medicine and Finance*. Taylor & Francis. ISBN: 9781420033397.

Marek Barwiński (2004). „Product-based metric for Gabor functions and its implications for the matching pursuit algorithm". Prac. mag. Uniwersytet Warszawski.

Christian G. Bénar i in. (2009). „Consensus Matching Pursuit for multi-trial EEG signals". W: *Journal of Neuroscience Methods* 180.1, s. 161–170. ISSN: 0165-0270. DOI: **10.1016/j.jneumeth.2009.03.005**.

Shengqin Bian i Lixin Zhang (2021). „Overview of Match Pursuit Algorithms and Application Comparison in Image Reconstruction". W: *2021 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)*, s. 216–221. DOI: **10.1109/IPEC51340.2021.9421295**.

Katarzyna J. Blinowska i Jarosław Żygierewicz (2022). *Practical Biomedical Signal Analysis Using MATLAB*. 2nd. USA: CRC Press, Inc. ISBN: 978-1-138-36441-7.

Tommaso Cai i Lie Wang (2011). „Orthogonal Matching Pursuit for Sparse Signal Recovery With Noise". W: *IEEE Transactions on Information Theory* 57, s. 4680–4688. DOI: **10.1109/TIT.2011.2146090**.

Scott Shaobing Chen, David L. Donoho i Michael A. Saunders (2001). „Atomic Decomposition by Basis Pursuit". W: *SIAM Review* 43.1, s. 129–159. DOI: **10.1137/S003614450037906X**.

Yuejie Chi i Robert Calderbank (2012). „Coherence-based performance guarantees of Orthogonal Matching Pursuit". W: *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, s. 2003–2009. DOI: **10.1109/Allerton.2012.6483468**.

James Cooley i John Tukey (1965). „An Algorithm for the Machine Calculation of Complex Fourier Series". W: *Mathematics of Computation* 19.90, s. 297–301.

Ingrid Daubechies (1992). *Ten Lectures on Wavelets*. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial i Applied Mathematics. ISBN: 9780898712742.

David L. Donoho i in. (2012). „Sparse Solution of Underdetermined Systems of Linear Equations by Stagewise Orthogonal Matching Pursuit". W: *IEEE Transactions on Information Theory* 58, s. 1094–1121. DOI: **10.1109/TIT.2011.2173241**.

Piotr J. Durka (2007). *Matching Pursuit and Unification in EEG analysis*. Engineering in Medicine and Biology. ISBN 978-1-58053-304-1. Norwood, MA: Artech House.

Piotr J. Durka i Katarzyna J. Blinowska (1995). „Analysis of EEG transients by means of matching pursuit". W: *Annals of Biomedical Engineering* 23, s. 608–611. DOI: **10.1007/BF02584459**.

Piotr J. Durka, Marian Dovgialo i in. (2024). „Two-Stage Atomic Decomposition of Multichannel EEG and the Previously Undetectable Sleep Spindles". W: *Sensors* 24.3. ISSN: 1424-8220. DOI: **10.3390/s24030842**.

Piotr J. Durka, Dobiesław Ircha i Katarzyna J. Blinowska (2001). „Stochastic time-frequency dictionaries for matching pursuit". W: *IEEE Transactions on Signal Processing* 49.3, s. 507–510. DOI: **10.1109/78.905866**.

Piotr J. Durka, Edward F. Kelly i Katarzyna J. Blinowska (1996). „Time-frequency analysis of stimulus-driven EEG activity by matching pursuit". W: *Proceedings of 18th Annual International Conference of the IEEE Engineering in Medicine and Biology Society* 3, 1009–1010 vol.3. DOI: **10.1109/IEMBS.1996.652684**.

Piotr J. Durka, Urszula Malinowska i in. (2015). „Spindles in Svarog: framework and software for parametrization of EEG transients". W: *Frontiers in Human Neuroscience* 9, s. 258. ISSN: 1662-5161. DOI: **10.3389/fnhum.2015.00258**.

Piotr J. Franaszczuk i in. (1998). „Time-frequency analysis using the matching pursuit algorithm applied to seizures originating from the mesial temporal lobe." W: *Electroencephalography and clinical neurophysiology* 106 6, s. 513–21. DOI: **10.1016/S0013-4694(98)00024-8**.

Matteo Frigo i Steven G. Johnson (2005). „The Design and Implementation of FFTW3". W: *Proceedings of the IEEE* 93.2. Special issue on "Program Generation, Optimization, and Platform Adaptation", s. 216–231.

Katerina Giannakaki i in. (2019). „Automatic Absence Seizure Detection Evaluating Matching Pursuit Features of EEG Signals". W: *2019 IEEE 19th International Conference on Bioinformatics and Bioengineering (BIBE)*, s. 886–889. DOI: **10.1109/BIBE.2019.00165**.

Michael M. Goodwin (1997). „Matching pursuit with damped sinusoids". W: *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing* 3, 2037–2040 vol.3. DOI: **10.1109/ICASSP.1997.599345**.

Justyna Gula i in. (2021). „Characterizing the middle-age neurophysiology using EEG/MEG". W: *bioRxiv*. DOI: **10.1101/2021.06.04.447084**.

Yousef Hannawi i in. (2015). „Resting brain activity in disorders of consciousness". W: *Neurology* 84.12, s. 1272–1280. DOI: **10.1212/WNL.0000000000001404**.

M. Jörn i in. (2011). „Single-trial reconstruction of auditory evoked magnetic fields by means of Template Matching Pursuit". W: *Journal of Neuroscience Methods* 199.1, s. 119–128. ISSN: 0165-0270. DOI: **10.1016/j.jneumeth.2011.04.019**.

Sreejith Kallummil i Sheetal Kalyani (2017). *Tuning Free Orthogonal Matching Pursuit*. arXiv: **1703.05080 [stat.ML]**.

Güneş Karabulut (2006). „Matching pursuit algorithms with applications in wireless communication systems". Prac. dokt. University of Ottawa. DOI: **10.20381/RUOR-12911**.

Sacha Krstulovic i Rémi Gribonval (maj 2006). „MPTK: Matching Pursuit made Tractable". W: *Proc. Int. Conf. Acoust. Speech Signal Process. (ICASSP'06)*. T. 3. Toulouse, France, s. III-496 –III-499.

Rafał Kuś, Piotr T. Różański i Piotr J. Durka (wrz. 2013). „Multivariate matching pursuit in optimal Gabor dictionaries: theory and software with interface for EEG/MEG via Svarog". W: *BioMedical Engineering OnLine* 12.1, s. 94. ISSN: 1475-925X. DOI: **10.1186/1475-925X-12-94**.

Carlos A. Loza (2019). „RobOMP: Robust variants of Orthogonal Matching Pursuit for sparse representations". W: *PeerJ Computer Science* 5. DOI: **10.7717/peerj-cs.192**.

Carlos A. Loza i José C. Príncipe (2016). „Transient model of EEG using Gini Index-based matching pursuit". W: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, s. 724–728. DOI: **10.1109/ICASSP.2016.7471770**.

Urszula Malinowska i in. (2007). „Explicit parameterization of sleep EEG transients". W: *Computers in biology and medicine* 37 4, s. 534–41. DOI: **10.1016/J.COMPBIOMED.2006.08.005**.

Stéphane G. Mallat i Zhifeng Zhang (1993). „Matching pursuits with time-frequency dictionaries". W: *IEEE Transactions on Signal Processing* 41.12, s. 3397–3415. DOI: **10.1109/78.258082**.

Ernst Niedermeyer i Fernando Lopez da Silva (2005). *Electroencephalography: Basic Principles, Clinical Applications, and Related Fields*. LWW Doody's all reviewed collection. Lippincott Williams & Wilkins. ISBN: 9780781751261.

NVIDIA, Péter Vingelmann i Frank H.P. Fitzek (2020). *CUDA, release: 10.2.89*. URL: **https://developer.nvidia.com/cuda-toolkit**.

Piotr T. Różański (2018). „Normalization effects in matching pursuit algorithm with Gabor dictionaries". W: *Journal of Applied Computer Science* 26.2, s. 187–199. ISSN: 1507-0360. URL: **https://it.p.lodz.pl/file.php/12/2018-2/jacs-2-2018-rozanski.pdf**.

Piotr T. Różański (2020). „Effects of envelope and dictionary structure on the performance of matching pursuit". W: *IET Signal Processing* 14.2, s. 89–96. DOI: **10.1049/iet-spr.2019.0246**.

Piotr T. Różański (2024). „*empi*: GPU-accelerated matching pursuit with continuous dictionaries". W: *ACM Trans. Math. Softw.* (przyjęte do druku).

Ron Rubinstein, Alfred M. Bruckstein i Michael Elad (2010). „Dictionaries for Sparse Representation Modeling". W: *Proceedings of the IEEE* 98.6, s. 1045–1057. DOI: **10.1109/JPROC.2010.2040551**.

Qin Shu-ren (2003). „Using Matching Pursuit Algorithm to Analyze Electroencephalogram (EEG) Sleep Spindles in Time-Frequency Domain". W: *Journal of Shanghai Jiaotong University*.

Joel A. Tropp i Anna C. Gilbert (2007). „Signal Recovery From Random Measurements Via Orthogonal Matching Pursuit". W: *IEEE Transactions on Information Theory* 53, s. 4655–4666. DOI: **10.1109/TIT.2007.909108**.

Jian Wang, Seokbeop Kwon i Byonghyo Shim (2011). „Generalized Orthogonal Matching Pursuit". W: *IEEE Transactions on Signal Processing* 60, s. 6202–6216. DOI: **10.1109/TSP.2012.2218810**.

Yinghua Wu i Victor S. Batista (2003). „Matching-pursuit for simulations of quantum processes". W: *Journal of Chemical Physics* 118, s. 6720–6724. DOI: **10.1063/1.1560636**.

Wu-yang Xiao i Xu-yuan Zheng (2012). „Time-frequency analysis based on matching pursuit and its application in epilepsy EEG". W: *International Journal of Biomedical Engineering* 35, s. 8–13. DOI: **10.3760/CMA.J.ISSN.1673-4181.2012.01.003**.

Magdalena Zieleniewska i in. (2019). „Parametric Description of EEG Profiles for Assessment of Sleep Architecture in Disorders of Consciousness". W: *International Journal of Neural Systems* 29.03. PMID: 30642201, s. 1850049. DOI: **10.1142/S0129065718500491**.

Jarosław Żygierewicz i in. (1998). „Time-frequency analysis of vibrotactile driving responses by matching pursuit". W: *Journal of Neuroscience Methods* 81.1, s. 121–129. ISSN: 0165-0270. DOI: **10.1016/S0165-0270(98)00016-8**.

# Artykuł A: *Multivariate matching pursuit in optimal Gabor dictionaries: theory and software with interface for EEG/MEG via Svarog*

Rafał Kuś, Piotr T. Różański i Piotr J. Durka (wrz. 2013). „Multivariate matching pursuit in optimal Gabor dictionaries: theory and software with interface for EEG/MEG via Svarog". W: *BioMedical Engineering OnLine* 12.1, s. 94. ISSN: 1475-925X. DOI: **10.1186/1475-925X-12-94**

Kuś *et al. BioMedical Engineering OnLine* 2013, **12**:94
http://www.biomedical-engineering-online.com/content/12/1/94

BioMedical Engineering OnLine

RESEARCH Open Access

# Multivariate matching pursuit in optimal Gabor dictionaries: theory and software with interface for EEG/MEG via Svarog

Rafał Kuś, Piotr Tadeusz Różański and Piotr Jerzy Durka*

*Correspondence:
durka@fuw.edu.pl
Faculty of Physics, University of Warsaw, ul. Hoża 69, 00-681 Warszawa, Poland

**Abstract**

**Background:** Matching pursuit algorithm (MP), especially with recent multivariate extensions, offers unique advantages in analysis of EEG and MEG.

**Methods:** We propose a novel construction of an optimal Gabor dictionary, based upon the metrics introduced in this paper. We implement this construction in a freely available software for MP decomposition of multivariate time series, with a user friendly interface via the Svarog package (Signal Viewer, Analyzer and Recorder On GPL, http://braintech.pl/svarog), and provide a hands-on introduction to its application to EEG. Finally, we describe numerical and mathematical optimizations used in this implementation.

**Results:** Optimal Gabor dictionaries, based on the metric introduced in this paper, for the first time allowed for *a priori* assessment of maximum one-step error of the MP algorithm. Variants of multivariate MP, implemented in the accompanying software, are organized according to the mathematical properties of the algorithms, relevant in the light of EEG/MEG analysis. Some of these variants have been successfully applied to both multichannel and multitrial EEG and MEG in previous studies, improving preprocessing for EEG/MEG inverse solutions and parameterization of evoked potentials in single trials; we mention also ongoing work and possible novel applications.

**Conclusions:** Mathematical results presented in this paper improve our understanding of the basics of the MP algorithm. Simple introduction of its properties and advantages, together with the accompanying stable and user-friendly Open Source software package, pave the way for a widespread and reproducible analysis of multivariate EEG and MEG time series and novel applications, while retaining a high degree of compatibility with the traditional, visual analysis of EEG.

**Keywords:** Matching pursuit, EEG, MEG, Time-frequency, Gabor dictionary, Metrics

## Background

Since the first application to EEG in 1995 [1], matching pursuit algorithm (MP) has been shown to significantly improve the EEG/MEG analysis in a variety of paradigms, including pharmaco-EEG [2,3], assessment of propagation [4], dynamics [5] and signal complexity [6-8] in epileptic seizures, detection of seizures [9,10], analysis of somatosensory evoked potentials in humans [11] and rats [12], detection of sleep spindles in Obstructive Sleep Apnea [13] and investigation of their chirping properties [14], studies of high gamma in

humans [15] and monkeys [16], investigation of brain's pain processing [17,18], parameterization of vibrotactile driving responses [19] and event-related desynchronization and synchronization [20,21].

New area of applications opened with the advent of multivariate MP (MMP) algorithms. MMP preprocessing was shown to significantly improve stability and sensitivity of EEG inverse solutions [22-27] and allowed for tracing evoked responses in single trials of EEG and MEG [28-31].

Finally, the algorithm offers also unique compatibility with the traditional, visual analysis of EEG. Specific mode of operation of MP, which is sequential focusing on locally strongest ("most visible") signal structures, resembles the working of an electroencephalographer who visually evaluates the EEG time series. Proper interpretation of the MP parameterization can provide a direct link to the results of visual analysis of EEG [32]—that is, we may find a direct correspondence between the waveforms fitted to the EEG time series, and the structures marked by visual scorer, including sleep spindles, slow waves or epileptic spikes [33-35]. This advantage should not be underestimated in the field, where most of our knowledge about behavioral and neurological correlates of EEG comes from visual analysis, which is still the only golden standard and point of reference: according to the Report of the American Academy of Neurology and the American Clinical Neurophysiology Society [36], quantitative EEG analysis *should be used only by physicians highly skilled in clinical EEG, and only as an adjunct to and in conjunction with traditional EEG interpretation*.

In spite of all these advantages, PubMed search of "matching pursuit and EEG" still returns only a two-digit number. This limited acceptance of such a promising method may be partly due to the lack of:

1. Well defined criteria for setting the most important parameters of the algorithm, which are the number and distribution of dictionary's functions.
2. Common framework for a variety of multivariate MP algorithms.
3. Robust and user friendly software based upon solid mathematical foundations.

Following sections address these issues from two different angles. Next section *An interactive tour of matching pursuit* provides plain English introduction of the major aspects and parameters of the algorithm, based on example computations. Following sections use equations to introduce optimal sampling of Gabor dictionaries and a common framework for multivariate MP (MMP), and Appendix discusses major numerical and mathematical optimizations used in the MMP implementation accompanying this paper.

## An interactive tour of matching pursuit

[Additional file 1] is a video tutorial for downloading and configuration of the Svarog package, used for computations and visualization of results (Figures 1, 2, 3, 4, 5 and 6 are screenshots from Svarog). Screencast in [Additional file 2] shows the steps from loading the signal to displaying interactive map of time-frequency energy distribution. [Additional file 3] contains help of the MP module from Svarog. Complete software environment used for computations presented in this paper (including examples from the following sections) is freely available from http://braintech.pl/svarog, as described in section *Software availability and requirements*.

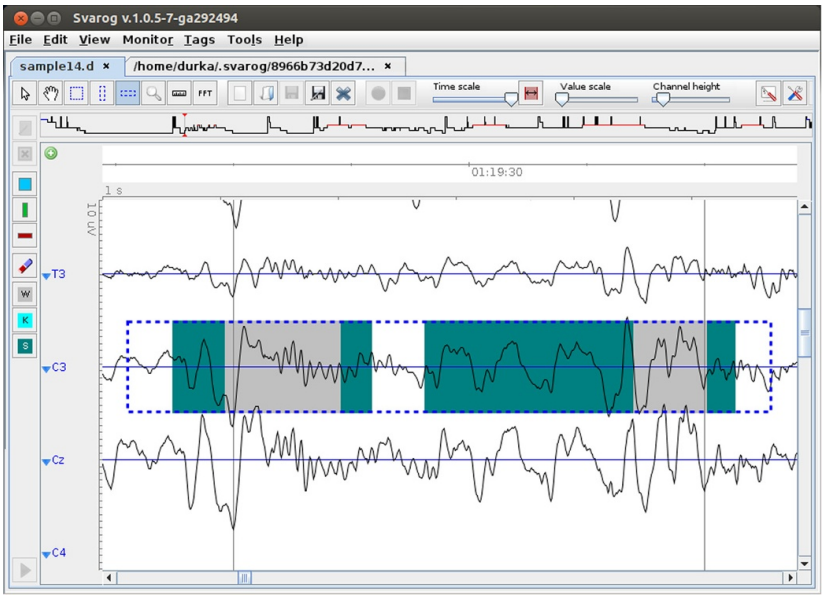


**Figure 1 Sample epoch of sleep EEG.** Screenshot displaying an epoch of sleep EEG recording. SWA and sleep spindles were marked by an electroencephalographer as green and gray rectangles, correspondingly (in this case both spindle tags fall inside the epochs marked as SWA). Blue dashed line outlines the epoch from C3 selected for MP decomposition, shown in Figure 2.

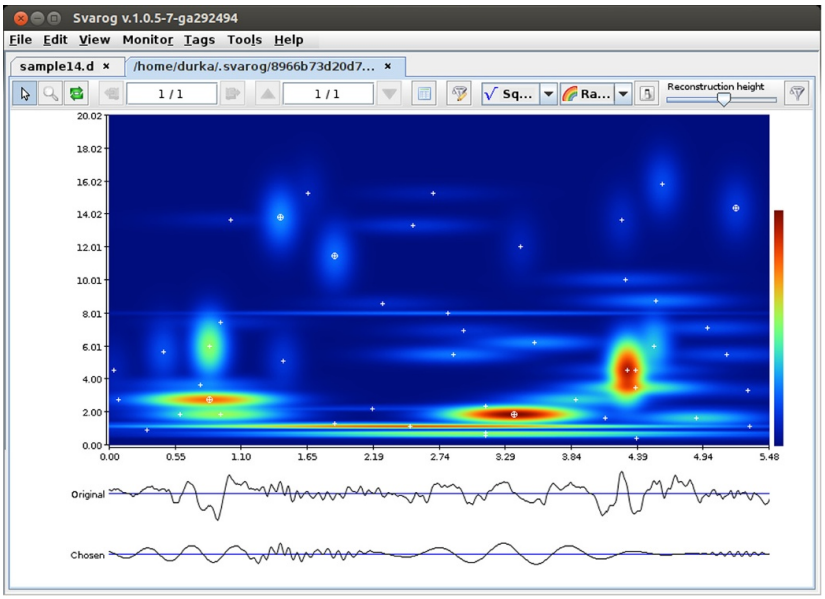


**Figure 2 Time-frequency distribution of signal's energy.** Results of MP decomposition displayed as an interactive time-frequency of map signal's energy density in Svarog. Clicking center of a blob (marked by white cross) adds the corresponding function to the reconstruction (bottom signal).

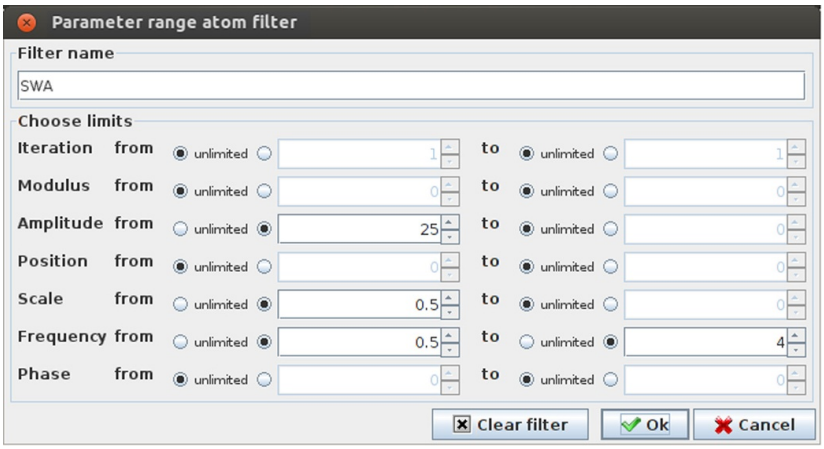


**Figure 3 Defining a filter for SWA.** Filter defining criteria for waveforms corresponding to SWA in the Svarog interface to MP.

### The algorithm

The gist of the matching pursuit algorithm can be summarized as follows:

1. We start by creating a huge, redundant set (called a dictionary) of candidate waveforms for representation of structures possibly occurring in the signal. For the time-frequency analysis of signals we use dictionaries composed of sines with Gaussian envelopes, called Gabor functions, which reasonably represent waxing and waning of oscillations.

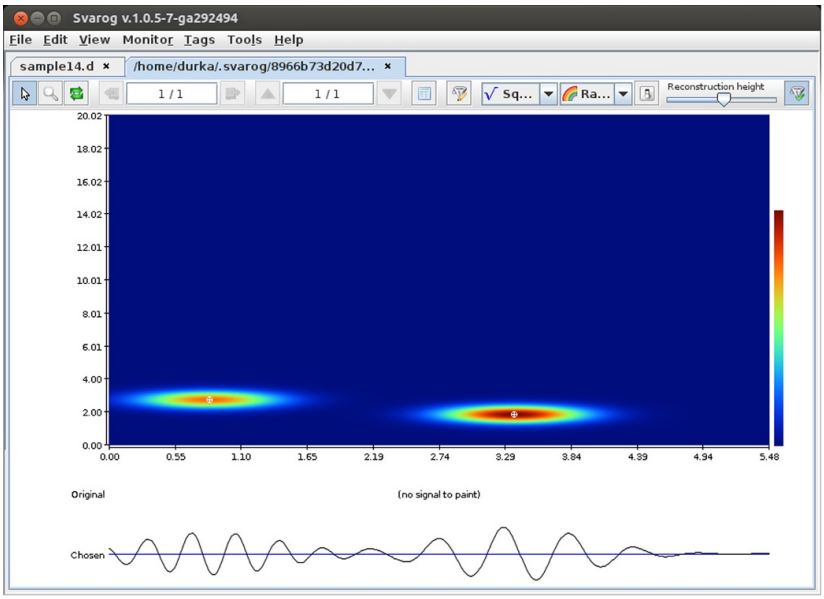


**Figure 4 Structures corresponding to SWA.** Result of the application of the filter from Figure 3 to the decomposition from Figure 2.

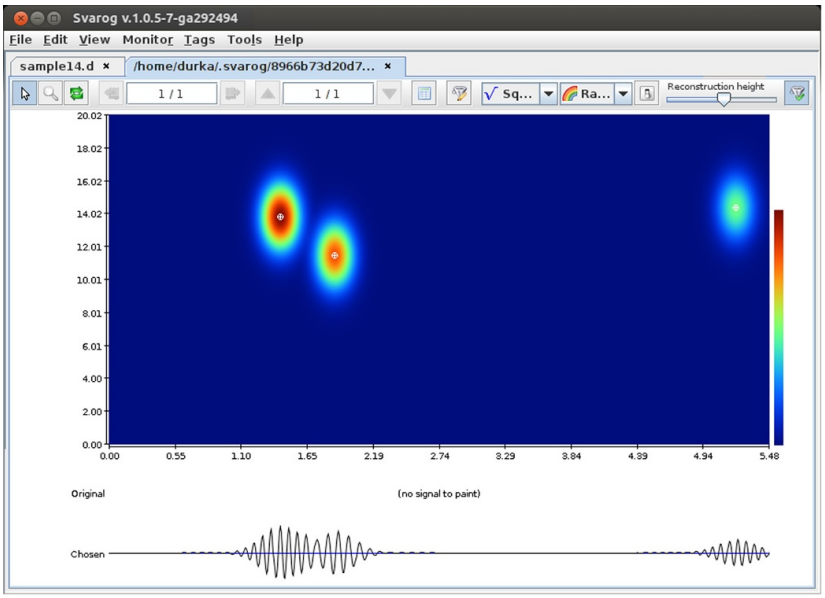


**Figure 5 Structures corresponding to sleep spindles.** Result of application of the filter defining sleep spindles to the decomposition from Figure 2.

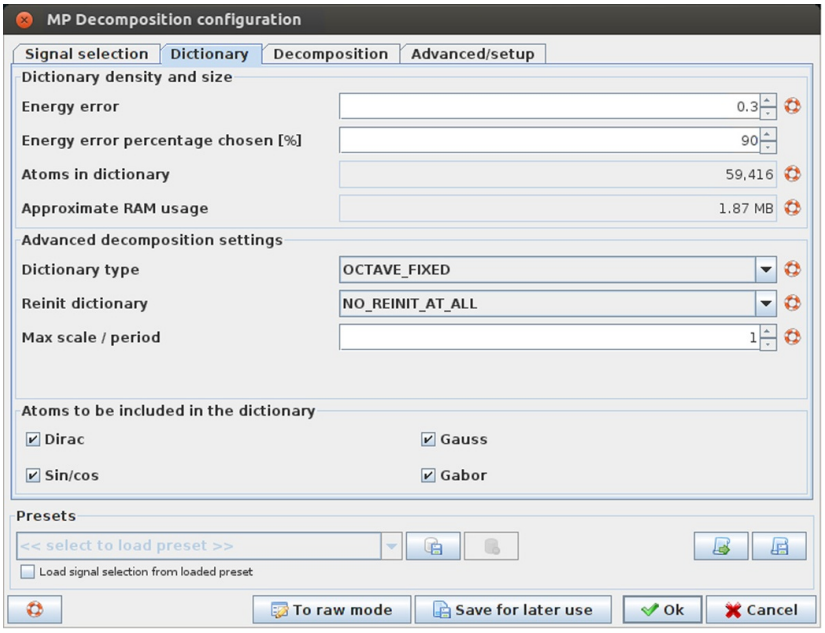


**Figure 6 Setting the dictionary parameters for MP decomposition in Svarog.** Tab for setting the parameters governing construction of the Gabor dictionary for MP decomposition in Svarog.

2. From this dictionary we choose only those functions, which fit the local signal structures. In such a way, the width of the analysis window is adjusted to the local properties of the signal. Local adaptivity of the procedure is somehow similar to the process of visual analysis, where an expert tends to separate local structure and assess their characteristics. Owing to this local adaptivity, MP is the only signal processing method returning explicit time span of detected structures.
3. The above idea is implemented in an iterative procedure: in each step we find one "best" function, and then subtract it from the signal being decomposed in the following steps.

**MP advantages in EEG**

We will discuss some of the advantages of MP in EEG analysis using an example from the field of sleep research. As mentioned in section *Background*, visual analysis of EEG is still the golden standard; in the area of sleep the basic reference [37] comes from 1968 (with later updates [38]). It defines criteria for division of sleep into stages, based mostly upon presence/prevalence of certain structures in the corresponding epochs of EEG recording. As for the definitions of these structures, formulated for standardization of their visual detection, let us take the example of sleep spindles:

> *The presence of sleep spindle should not be defined unless it is at least 0.5 sec. duration, i.e. one should be able to count 6 or 7 distinct waves within the half-second period. (...) The term should be used only to describe activity between 12 and 14 cps* [37].

Over the years, common definition drifted towards frequencies from 11 to 15 Hz, duration 0.5–2 seconds and amplitude usually above 15 $\mu V$. Reading this definition after 45 years, we are still surprised that:

1. Criteria are defined almost explicitly in time-frequency terms.
2. Before 1993 (that is before the introduction of the matching pursuit [39]), no signal processing method returning explicitly the frequency, amplitude and time width of the oscillations present in a signal was known.

In the following we present MP decomposition of an EEG epoch containing sleep spindles and slow wave activity (SWA). Figure 1 presents a sample epoch of sleep EEG loaded into Svarog (Signal Viewer, Analyzer and Recorder on GPL, see section *Software availability and requirements*). Green and gray rectangles represent SWA and sleep spindles, marked visually by an encephalographer.

Figure 2 presents results of MP decomposition of the epoch selected in Figure 1. Curves below the time-frequency map of energy density represent:

1. Original signal chosen for decomposition (marked in Figure 1).
2. Reconstruction (time course) of selected functions (marked by circled crosses).

Central panel holds the time-frequency map of signals energy density. Each function fitted to the signal by MP is presented as a Gaussian blob in the appropriate time and frequency coordinates, with time and frequency widths corresponding to its parameters. Formula for computing this representation is given by Eq. (4). One can argue about the superior properties of this estimate; indeed, there is a lot of possible ways to estimate signals energy density in the time-frequency plane (spectrogram, wavelets, Wigner-Ville etc.)

and none of them is perfect for all signals. Therefore, we shall concentrate on a unique feature of the MP decomposition, which is an explicit parameterization of the transients present in a signal.

Each of the blobs in Figure 2 represents a Gabor function of known time position and width, amplitude, frequency and phase, as in Eq. (3). These parameters contain the primary information about the signal's content (Eq. (2)), and they can be used directly for the investigation of the properties of the signal, as exemplified below.

Let us define the occurrence of SWA as a wave of amplitude above 50 $\mu V$, width above half a second and frequency from 0.5 to 4 Hz. Definition of such filter in Svarog is presented in Figure 3. The amplitude is entered as 25 $\mu V$, because the encephalographic convention relates to the peak-to-peak amplitude, which for the Gabor function is double of the mathematical amplitude. This convention is not the only difference between mathematics and visual perception of structures in the signal. For example, visual perception of both amplitude and time width depends on the context (mostly the variance of the signal). Therefore, if exact replication of the visual detection is the main goal, factors like S/N ratio can be incorporated into the post processing criteria.

Result of the application of the filter from Figure 3 to the decomposition from Figure 2 is given in Figure 4. Automagically, all that is left are structures conforming to the definition of SWA. Two major advantages of this result over previously available methods should be noted:

1. This selection takes into account all the features defining SWA, not only their frequency content as would be the case e.g. for a bandpass filter.
2. We have separate parameterization of all the conforming structures, including e.g. the duration of each of them.

The latter feature was explored in [40] for detection of sleep stages III and IV. These stages are defined as epochs occupied by SWA in 20–50% and over 50% of their time, correspondingly. As the previously employed signal processing methods did no parametrize explicitly the time span of relevant activity, it was the first explicit implementation of this rule, designed for standardization of visual detection, in a fully automatic system. We observe also a good concordance of these results with visual detection, represented by the two green marks (intersected by the gray marks for spindles) representing visually tagged SWA in Figure 1. This concordance was quantitatively evaluated in [34].

Similar operation can be performed for sleep spindles; application a filter reflecting their time-frequency parameters, quoted at the beginning of this section, gives us Figure 5. Again, structures detected by the algorithm fall within the borders marked by enecephalographer for spindles (gray tags in Figure 1). We observe also a typical example, when two superimposed spindles (on the left) were marked by the human expert as one. It exemplifies the fact that MP-based methods are in most cases downward compatible with visual detection, yet, apart from repeatability and automatization offer also increased sensitivity and resolution. Indeed, in the noisy signal it is almost impossible to see the boundary, and human brain did not evolve for an online calculation of instantaneous frequency, which differentiates these two spindles.

Similarly to the detection of SWA, this scheme is not only more sensitive and selective compared to previous approaches, while retaining a high degree of concordance with

visual detection [33,41], but also allows us, for example, to explicitly count the number of spindles occurrences in any given epoch—a parameter also relevant in sleep analysis.

Finally, it is worth noting that the above proposed procedure is essentially free from any method-dependent settings, like the choice of the mother wavelet in wavelet transform, window width in spectrogram, or smoothing kernel in Wigner-Ville derived representations. All these parameters can significantly alter the results of analysis, and optimal settings can be different for each analyzed epoch. On the contrary, matching pursuit decomposition as such is "more or less" uniquely defined by Equation (1). However, subtle relations between dictionaries and results of MP decomposition were not fully understood so far—their explanation constitutes the main mathematical result presented in this paper. So for the rest of this section let us concentrate on the "more or less".

#### Parameters of MP decomposition

As introduced in section *MP algorithm*, MP searches for functions fitting the signal in a large and redundant set called dictionary. The bigger (more redundant) this dictionary is, the better the chance of a perfect fit. Although it's a vague statement, nothing substantially more precise had been said on the relation of the dictionary size and quality of MP decomposition—until now.

This study introduces a novel construction of the dictionary, in which the inner products of the adjacent atoms are kept constant. In other words, distribution of dictionary's functions is uniform with respect to the metric related to their inner product. Using this construction gives us control over the maximum error of a single MP iteration, measured in terms of energy (product of the signal and a function from the optimal dictionary). Upper bound on this error is given by the worst case, where a structure present in the signal falls "in the middle" between dictionary's functions available for decomposition. This error is not equivalent to the resolution of approximation, because MP is an iterative, nonlinear procedure. Formal introduction of the mentioned metric and construction of the optimal dictionary are given in section *Optimal sampling of Gabor dictionaries*.

Figure 6 presents the "Dictionary" tab from the MP configuration window. The most important parameter (first from the top) is called "Energy error". It corresponds to $\epsilon$ from Equation (7) in section *Optimal sampling of Gabor dictionaries*, and relates directly to the mentioned above maximum error of a single MP iteration.

On the practical side, this parameter regulates the price/performance tradeoff in MP decomposition. Smaller (closer to zero) values result in larger amount of functions in the dictionary and higher resolution of resulting MP decomposition, at a price of increased computation times and memory requirements. After each change of this parameter the window displays the amount of RAM that will be occupied by the corresponding dictionary. Keeping "Energy error" constant for analysis of epochs of different sizes will result in larger dictionaries for longer epochs, but the accuracy of decomposition, related to the density of the dictionary, will be the same except for the border effects.

Another issue related to the dictionary is a possible statistical bias, present in decompositions of many epochs in the same, relatively small dictionary. This problem was discussed in [42], where solution in terms of stochastic dictionaries was proposed. Stochastic dictionaries introduced in [42] were constructed by explicit random drawing of functions parameters from defined ranges (subsection *Uniform sampling*). This approach gave less control over the structure of the dictionary and excluded possibilities of several valuable

numerical optimizations. In this paper we propose a different approach, where randomization is achieved by random removal of a defined fraction of functions from a dense, structured dictionary. This procedure is applied when user chooses "OCTAVE_STOCH" as "Dictionary type". In such case, a dictionary is first created according to the parameter 'Energy error" ($\epsilon$ from Equation (7)), and then selected fraction of functions is randomly removed from the dictionary. This fraction is governed by the parameter "Percentage chosen", so that (100 − Percentage chosen)% of functions is removed.

Finally, the last of the major parameters sets the number of matching pursuit iterations, which is equivalent to the number of functions chosen for the representation (compare Equation (2)). Owing to the convergence property of MP, they are ordered by decreasing energy. Increasing the number of iterations does not influence the parameters of functions chosen in earlier steps.

That is, if we make two separate decompositions of the same signal epoch and using the same dictionary—the first one with 50 iterations and the second one with 10 iterations—then the result of the first decomposition will be the same as taking the first 10 functions from the second decomposition (however, in case of stochastic dictionaries, these two decompositions will be performed using slightly different dictionaries). This difference is more pronounced for smaller dictionaries.

There are several mathematical criteria for stopping the decomposition; their influence was evaluated e.g. for MP-based descriptors of signal complexity [6] or optimal video coding [43]—general discussion of this parameter can be found in [44]. Software presented in this paper implements the two most basic options, based upon the maximum number of iterations ("Max iterations") and percentage of signal's energy explained by the whole decomposition ("Energy percent"). These options work in logical conjunction. Hence e.g. for the default settings (99% and 50 iterations), the procedure is completed after the 50th iteration—or before, if the representation (2) explains 99% of energy of the original signal for a lower number of functions $M$.

Detailed information on the parameters of MP decomposition employed in mp5 is available from [Additional file 3], which contains the help of the mp5 module from Svarog.

### Optimal sampling of Gabor dictionaries

We start by formally reintroducing the MP following the notation from the seminal paper by Mallat and Zhang [39]. Denoting the function fitted to the signal $x$ in the $n$-th iteration of MP as $g_{\gamma_n}$, and $n$-th residual as $R^n x$, we define the procedure as:

$$\begin{cases} R^0 x = x \\ R^n x = \langle R^n x, g_{\gamma_n} \rangle g_{\gamma_n} + R^{n+1} x \\ g_{\gamma_n} = \arg\max_{g_{\gamma_i} \in D} |\langle R^n x, g_{\gamma_i} \rangle| \end{cases} \tag{1}$$

As a result we get an expansion

$$x \approx \sum_{n=0}^{M-1} \langle R^n x,\ g_{\gamma_n} \rangle g_{\gamma_n} \tag{2}$$

where $M$ equals the number of iterations. For a time-frequency analysis of real-valued signals, dictionary $D$ is usually composed from Gabor functions

$$g_\gamma(t) = K(\gamma) e^{-\pi \left(\frac{t-u}{s}\right)^2} \cos\left(\omega(t-u) + \phi\right) \tag{3}$$

where $\gamma = (u, \omega, s)$ and $K(\gamma)$ is such that $||g_\gamma|| = 1$.

From expansion (2) we can derive a time-frequency distribution of the signal's energy (as shown in Figure 2), by summing Wigner distributions $W$ of selected functions and explicitly omitting cross-terms:

$$Ex(t,\omega) = \sum_{n=0}^{M} \left|\langle R^n x,\ g_{\gamma_n}\rangle\right|^2\ \ Wg_{\gamma_n}(t,\omega) \tag{4}$$

This magnitude is presented in Figures 2, 4 and 5.

#### Real world implementations of Gabor dictionaries

Parameters $(u,\omega,s)$ of functions $g_\gamma$ from Equation (3) form a 3-dimensional continuous subspace of $\mathbb{R}^3$—the infinite set $D_\infty$. Ranges of parameters delimiting this subspace correspond to the usual assumptions that the time center $u$ and time spread $s$ do not exceed the length of the analyzed epoch, and to the fact that $\omega$ makes no sense above the Nyquist frequency. The fourth parameter $\phi$ does not have to be taken into account in the construction of the dictionary $D$, since for any $g_\gamma$ we can find the phase $\phi$ maximizing its product $\langle g_\gamma, x\rangle$ with given signal $x$ in a single step—we recall this method in the Appendix (section *Optimal phase of a Gabor function*).

From this $D_\infty$, a finite dictionary $D \subset D_\infty$ must be chosen for any practical implementation of MP. This choice is equivalent to the choice of a discrete and finite set $\gamma$ from the continuous ranges of $u$, $\omega$, and $s$. However, up to now the major questions, crucial for real-world implementations of MP, were unanswered:

1. How should we choose the elements of the finite dictionary $D$ from $D_\infty$ and why?
2. How much do we gain by increasing the size of the dictionary?

It has been a well known fact, that using a larger dictionary of functions for decomposition of the same signal should lead to more accurate parametrization. However, no direct relation between the density of the dictionary and resolution of the resulting decomposition was derived so far. While an exact measure of the resolution of the highly nonlinear matching pursuit remains an open question, in the following sections we relate the maximum error of a single MP iteration to the density of an optimally sampled Gabor dictionary, filled with Gabor functions distributed uniformly with respect to the metric related to their inner product.

#### Dyadic sampling

The first, wavelet-like subsampling of $D_\infty$ was proposed in the seminal paper [39]:

$$\gamma = (u,\omega,s) = \left(p\,a^j\Delta u,\ k\,\frac{\Delta\omega}{a^j},\ a^j\right) \tag{5}$$

where $j,k,p \in \mathbb{Z}$ and the basic intervals for time and frequency ($\Delta u$ and $\Delta\omega$) are chosen so that $\Delta u = \frac{\Delta\omega}{2\pi} < 1$. If a dictionary $D$ is constructed from the functions $g_\gamma$ with parameters $\gamma = (u,\omega,s)$ chosen according to (5), then for any signal $x \in L^2(\mathbb{R})$ there exists an optimality factor $\alpha > 0$ such that

$$\sup_{g_\gamma \in D} |\langle x, g_\gamma\rangle| \geq \alpha \sup_{g \in D_\infty} |\langle x, g\rangle| \tag{6}$$

However, there were no quantitative estimates for the performance of the MP in given dictionary, constructed for given $\Delta\omega$ and $a$, and it seems that nothing can be said about the optimality factor $\alpha$ except for that it exists and is greater than zero.

#### Uniform sampling

The quest for a justification of a particular scheme of subsampling $D_\infty$ leads us to the considerations of a uniform sampling. If we know that the parameters $(u, \omega, s)$ are uniformly sampled with steps $\Delta u$, $\Delta\omega$, and $\Delta s$, then for any $g \in D_\infty$ there exists $g_\gamma \in D$, which has the parameters differing at most by halves of the sampling steps. Such sampling provides an estimate of one-step resolution of the highly nonlinear MP algorithm in the space of the parameters of Gabor functions. It was implemented in the mp4 software package [45] as a byproduct of the first approach to the issue of stochastic dictionaries [42], where parameters of the dictionary's functions were drawn from uniform probability distributions.

However, such a sampling scheme in some cases is far from optimal for the MP algorithm. For example, constant sampling of the time interval $\Delta u$ for all the widths $s$ will result in a dictionary containing a lot of strongly overlapping Gabors with large widths $s$ (which will give large inner products with neighbors from the dictionary), and much more sparse coverage of positions of "shorter" Gabors with small $s$. Choice (1) of functions for the MP expansion (2) is based entirely on inner products, and the Cartesian metric in the space of the parameters of these functions is far from optimal in this respect. This leads us to the search for a metric that would correspond to the MP selection procedure (1).

#### In pursuit of a relevant metric

A metric that would reflect the distance between the dictionary elements as "seen" by the MP algorithm should be based on the inner product of dictionary's elements. We propose a metric $d(g_0, g_1)$ to ensure that the distance between nearest Gabor atoms in the dictionary does not exceed some *a priori* given threshold, $\epsilon$. Similarly to dyadic dictionary (see Eq. 5), we assume that the scale parameter $s$ varies by a factor $a$ (dilation factor), as follows: $s_j = a^j$, where $j > 0$ and $a > 1$ . However, we will propose a new method for determination of step values $\Delta\omega$ and $\Delta u$. We will show that this parametrization allows us to derive such step values, that for every $(s, \omega, u)$

$$\begin{aligned} d(g_{(s,\omega,u,\phi)}, g_{(as,\omega,u,\phi)}) &\le \epsilon \\ d(g_{(s,\omega,u,\phi)}, g_{(s,\omega+\Delta\omega,u,\phi)}) &\le \epsilon \\ d(g_{(s,\omega,u,\phi)}, g_{(s,\omega,u+\Delta u,\phi)}) &\le \epsilon\,. \end{aligned} \tag{7}$$

In order to construct a proper metric $d(g_0, g_1)$, we start by introducing a simple, intuitive measure for Gabor function similarity which satisfies all the properties of a metric:

$$d_0(g_0, g_1) = 2^{-\frac{1}{2}} \left\| g_0 - g_1 \right\| = 2^{-\frac{1}{2}} \sqrt{\langle g_0 - g_1 | g_0 - g_1 \rangle} = \sqrt{1 - \langle g_0 | g_1 \rangle}\,. \tag{8}$$

Norm $d_0$ above is naturally defined based upon the inner product in the space of the real signals. The constant $2^{-\frac{1}{2}}$ has been introduced to set the distance between orthogonal Gabor atoms to 1, while greater distances (up to 2) can exist between Gabor atoms whose inner products are negative.

Inner product of two Gabor functions has the dimension of energy, thus the dimension of the metric (8) is amplitude. According to this fact, the conditions (7) imply the dimension of the square of the parameter $\epsilon$ as the quantity of energy. Also, $\langle G_0 | G_1 \rangle_{\max}$ will be nonnegative for any two Gabor functions. Therefore, since Gabors are normalized, the value of their inner product is limited to the range $(0, 1)$ and thus the limit of the $\epsilon$ is

$(0, 1)$. However, $\epsilon = 1$ allows the neighboring atoms to be orthogonal, so for a reasonable MP decomposition one should require $\epsilon \ll 1$.

To the best of our knowledge, the above measure has not been previously applied to the Gabor dictionary construction problem, although a similar one for packing in projection spaces was introduced by Tropp [46]. Interpretation and possible values of parameter $\epsilon$ will be discussed in the following section, after we construct a final metric based on $d_0(g_0, g_1)$.

#### Phase-related equivalence

According to the section *Optimal phase of a Gabor function* from the Appendix, one can easily calculate phase $\phi$ of the Gabor function that maximizes its inner product with a given signal. Therefore, each atom in MP dictionary can be replaced with any other representative from the set $G_{(s,\omega,u)}$, which is an equivalence class in respect to a relation $\sim$ defined as

$$g_{(s_0,\omega_0,u_0,\phi_0)} \sim g_{(s_1,\omega_1,u_1,\phi_1)} \iff (s_0 = s_1) \wedge (\omega_0 = \omega_1) \wedge (u_0 = u_1)\,. \tag{9}$$

This feature has to be taken into account by introducing a correct metric for Gabor atoms

$$d(g_0, g_1) = \tilde{d}(g_0/\sim,\, g_1/\sim)\,, \tag{10}$$

where $\tilde{d}(G_0, G_1)$ is the distance function defined on equivalence classes

$$\tilde{d}(G_0, G_1) = \min_{\substack{g_0 \in G_0 \\ g_1 \in G_1}} d_0(g_0, g_1) = \sqrt{1 - \max_{\substack{g_0 \in G_0 \\ g_1 \in G_1}} \langle g_0 | g_1 \rangle} = \sqrt{1 - \langle G_0 | G_1 \rangle_{\max}}\,, \tag{11}$$

and we introduce the *maximal scalar product* $\langle G_0 | G_1 \rangle_{\max}$, which can be calculated as

$$\langle G_{(s_0,\omega_0,u_0)} | G_{(s_1,\omega_1,u_1)} \rangle_{\max} = \max_{\phi_0,\phi_1 \in [0;2\pi]} \langle g_{(s_0,\omega_0,u_0,\phi_0)} | g_{(s_1,\omega_1,u_1,\phi_1)} \rangle\,. \tag{12}$$

#### Inner products of Gabor functions

The measure $d_0$ (8) is based on an inner product of Gabor functions, so it is necessary to calculate analytical formulae for this product. Such expressions can be found for real Gabor functions

$$g_\gamma(t) = K_\gamma e^{-\pi \left(\frac{t-u}{s}\right)^2} \cos(\omega(t-u) + \phi)\,, \tag{13}$$

where $\gamma \equiv (s, \omega, u, \phi)$ is a set of the parameters, and

$$K_\gamma = \frac{2^{3/4}}{\sqrt{s}} \left[1 + \cos(2\phi) e^{-\frac{s^2\omega^2}{2\pi}}\right]^{-\frac{1}{2}} \tag{14}$$

is the normalization factor.

To present an analytical formula for the inner product of two real Gabor functions (13) $g_0$ and $g_1$ with normalization factors $K_0$ and $K_1$ respectively, we introduce following constants:

$$A = \pi \left(\frac{1}{s_0^2} + \frac{1}{s_1^2}\right) \tag{15}$$

$$B = \frac{\pi}{A} \left(\frac{u_0}{s_0^2} + \frac{u_1}{s_1^2}\right) \tag{16}$$

$$C = -\pi\left(\frac{u_0^2}{s_0^2} + \frac{u_1^2}{s_1^2}\right) + AB^2 = -\pi\frac{(u_0 - u_1)^2}{s_0^2 + s_1^2}. \tag{17}$$

The complete formula for the product has been adapted from [47]:

$$\begin{aligned}\langle g_0, g_1\rangle = K_0K_1\sqrt{\frac{\pi}{4A}}e^C\Bigg(&\cos\left[(\omega_0+\omega_1)B + (\phi_0+\phi_1) - (\omega_0u_0+\omega_1u_1)\right]e^{-\frac{(\omega_0+\omega_1)^2}{4A}} + \\ &+ \cos\left[(\omega_0-\omega_1)B + (\phi_0-\phi_1) - (\omega_0u_0-\omega_1u_1)\right]e^{-\frac{(\omega_0-\omega_1)^2}{4A}}\Bigg).\end{aligned} \tag{18}$$

#### Products of adjacent atoms

To simplify the inner product (18) for special cases of adjacent dictionary atoms, we will calculate distances between two atoms which differ in only one of the three parameters $s, \omega, u$. Let us discuss each case separately.

##### *Scale*

Because $u_0 = u_1$, the scalar product is invariant to a shift in position $u$, so we can safely choose $u = 0$. In this case $A = \frac{\pi(a^2+1)}{a^2s^2}$, $B = 0$, $C = 0$. Therefore,

$$\langle g_{(s,\omega,u=0,\phi_0)}|g_{(as,\omega,u=0,\phi_1)}\rangle = \sqrt{\frac{2a}{a^2+1}}\frac{\cos(\phi_0-\phi_1) + \cos(\phi_0+\phi_1)e^{-\frac{a^2s^2\omega^2}{(a^2+1)\pi}}}{\sqrt{1+\cos(2\phi_0)e^{-\frac{s^2\omega^2}{2\pi}}}\sqrt{1+\cos(2\phi_1)e^{-\frac{a^2s^2\omega^2}{2\pi}}}}. \tag{19}$$

The global maximum of this product will be present for $\phi_0 = \phi_1 = 0$ or $\phi_0 = \phi_1 = \frac{\pi}{2}$, depending on parameter values. Therefore, one can calculate

$$\langle G_{(s,\omega,u)}|G_{(as,\omega,u)}\rangle_{\max} = \sqrt{\frac{2a}{a^2+1}}\max\left\{\frac{1-e^{-\frac{a^2s^2\omega^2}{(a^2+1)\pi}}}{\sqrt{1-e^{-\frac{s^2\omega^2}{2\pi}}}\sqrt{1-e^{-\frac{a^2s^2\omega^2}{2\pi}}}}; \frac{1+e^{-\frac{a^2s^2\omega^2}{(a^2+1)\pi}}}{\sqrt{1+e^{-\frac{s^2\omega^2}{2\pi}}}\sqrt{1+e^{-\frac{a^2s^2\omega^2}{2\pi}}}}\right\}. \tag{20}$$

##### *Frequency*

The frequency parameter $\omega$ changes by $\Delta\omega$, $s_0 = s_1$ and $u_0 = u_1$. We can choose $u = 0$, analogically to the previous case. Here $A = \frac{2\pi}{s^2}$, $B = 0$ and $C = 0$, so the product is as follows:

$$\langle g_{(s,\omega,u=0,\phi_0)}|g_{(s,\omega+\Delta\omega,u=0,\phi_1)}\rangle = e^{-\frac{\Delta\omega^2s^2}{8\pi}}\frac{\cos(\phi_0-\phi_1) + \cos(\phi_0+\phi_1)e^{-\frac{s^2\omega(\omega+\Delta\omega)}{2\pi}}}{\sqrt{1+\cos(2\phi_0)e^{-\frac{s^2\omega^2}{2\pi}}}\sqrt{1+\cos(2\phi_1)e^{-\frac{s^2(\omega+\Delta\omega)^2}{2\pi}}}}. \tag{21}$$

The global maximum of this product will be present for $\phi_0 = \phi_1 = \frac{\pi}{2}$. Therefore,

$$\langle G_{(s,\omega,u)}|G_{(s,\omega+\Delta\omega,u)}\rangle_{\max} = e^{-\frac{\Delta\omega^2s^2}{8\pi}}\frac{1-e^{-\frac{s^2\omega(\omega+\Delta\omega)}{2\pi}}}{\sqrt{1-e^{-\frac{s^2\omega^2}{2\pi}}}\sqrt{1-e^{-\frac{s^2(\omega+\Delta\omega)^2}{2\pi}}}}. \tag{22}$$

***Position***

In this case, $s_0 = s_1$, $\omega_0 = \omega_1$, and position parameter $u$ differs by $\Delta u$. We can choose $u_0 = -\frac{\Delta u}{2}$ and $u_1 = \frac{\Delta u}{2}$. Applying $A = \frac{2\pi}{s^2}$, $B = 0$, $C = -\frac{\pi \Delta u^2}{2s^2}$ lead to

$$\langle g_{(s,\omega,u_0=-\frac{\Delta u}{2},\phi_0)} | g_{(s,\omega,u_1=\frac{\Delta u}{2},\phi_1)} \rangle = e^{-\frac{\pi \Delta u^2}{2s^2}} \frac{\cos(\phi_0 - \phi_1 + \omega \Delta u) + \cos(\phi_0 + \phi_1) e^{-\frac{s^2\omega^2}{2\pi}}}{\sqrt{1 + \cos(2\phi_0) e^{-\frac{s^2\omega^2}{2\pi}}} \sqrt{1 + \cos(2\phi_1) e^{-\frac{s^2\omega^2}{2\pi}}}}. \tag{23}$$

The global maximum of the product will appear for $\phi_0 = -\phi_1$. Therefore,

$$\langle G_{(s,\omega,u)} | G_{(s,\omega,u+\Delta u)} \rangle_{\max} = e^{-\frac{\pi \Delta u^2}{2s^2}} \max_{\phi \in [0;2\pi]} \frac{\cos(2\phi + \omega \Delta u) + e^{-\frac{s^2\omega^2}{2\pi}}}{1 + \cos(2\phi)\, e^{-\frac{s^2\omega^2}{2\pi}}}. \tag{24}$$

**Construction of the optimally sampled dictionary**

Formulae (20), (22) and (24) substituted into (7) could be used to construct an optimal dictionary. However, to improve computational performance of MP algorithms, one can use Fast Fourier Transform (FFT) as described in the Appendix. Therefore, it would be preferable to have values of $\Delta\omega$ and $\Delta u$ independent of $\omega$, yet still satisfying (7).

To construct such "uniform" dictionary, step values ($a$, $\Delta\omega$, $\Delta u$) must be selected for such $\omega$ that minimizes the value of the maximal scalar product (12). Exemplary step values obtained from (20), (22) and (24) are shown in Figure 7, as a function of frequency $\omega$, for three different values of parameter $\epsilon$. It can be observed that in order to obtain $a$, $\Delta\omega$, $\Delta u$ fulfilling the condition (7) and independent of $\omega$, one has to select step values corresponding to a large value of frequency $\omega$.

Moreover, at frequency $\omega \approx \pi$, one can safely substitute in Equations (20), (22) and (24):

$$\begin{aligned} e^{-\frac{s^2\omega^2}{2\pi}} &\approx 0 \\ e^{-\frac{a^2 s^2 \omega^2}{2\pi}} &\approx 0 \\ e^{-\frac{a^2 s^2 \omega^2}{(a^2+1)\pi}} &\approx 0 \\ e^{-\frac{s^2 \omega(\omega+\Delta\omega)^2}{2\pi}} &\approx 0\,. \end{aligned} \tag{25}$$

With these approximations, maximal scalar products can be calculated easily as

$$\begin{aligned} \langle G_{(s,\omega,u)} | G_{(as,\omega,u)} \rangle_{\max} &\approx \sqrt{\frac{2a}{a^2+1}} \\ \langle G_{(s,\omega,u)} | G_{(s,\omega+\Delta\omega,u)} \rangle_{\max} &\approx e^{-\frac{\Delta\omega^2 s^2}{8\pi}} \\ \langle G_{(s,\omega,u)} | G_{(s,\omega,u+\Delta u)} \rangle_{\max} &\approx e^{-\frac{\pi \Delta u^2}{2s^2}}, \end{aligned} \tag{26}$$

taking into account that $\max_{\phi \in [0;2\pi]} \cos(2\phi + \omega \Delta u) = 1$.

Therefore, the optimal step values independent of $\omega$ can be calculated from Equations (7) in the following steps:

1. For a given threshold $\epsilon$ (see Eq. 7) one can calculate the dilation factor $a$:

$$a = \frac{1 + \epsilon\sqrt{(2-\epsilon^2)(\epsilon^4 - 2\epsilon^2 + 2)}}{\left(1 - \epsilon^2\right)^2} \tag{27}$$

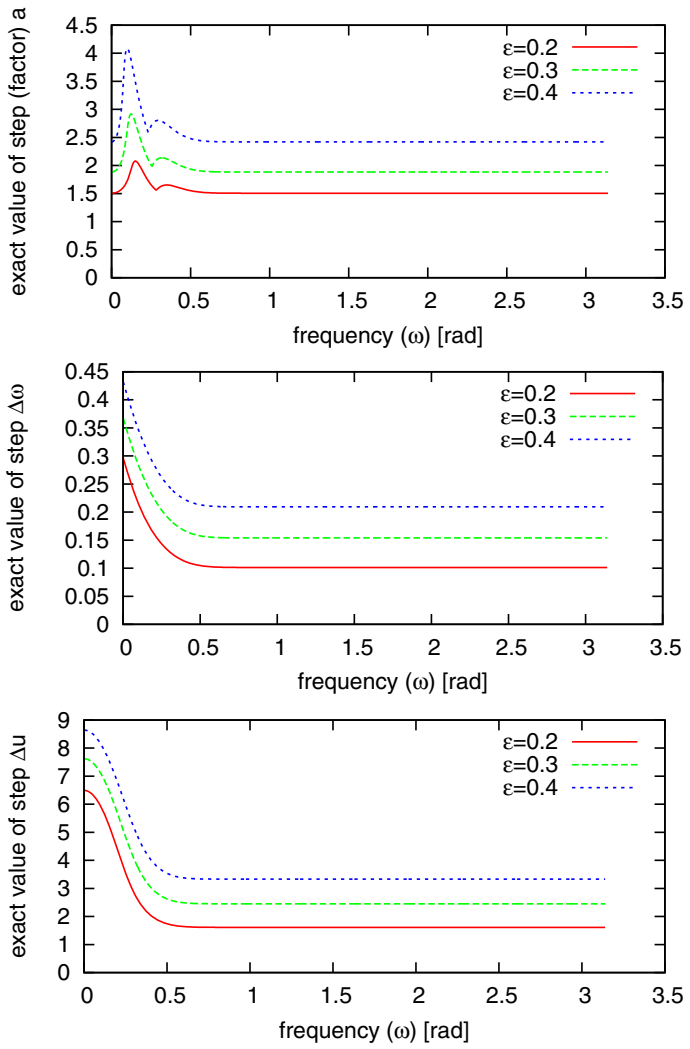


**Figure 7 Optimal scale factor, frequency step and position step.** Optimal scale factor $a$ (top plot—see Eq. (20)), frequency step $\Delta\omega$ (middle plot—see Eq. (22)) and position step $\Delta u$ (see Eq. (24)) as a function of $\omega$ for three values of $\epsilon$ and scale $s = 10$.

2. The set of scales $s_1, s_2, \ldots s_{N_s}$ of Gabors are determined as:

$$s_j = a^j, \tag{28}$$

where $j \in \mathbb{Z}_+$, $N_s = \lfloor \log_a N \rfloor$ is the number of different scales and $N$ is the length of the analyzed signal.

3. For each scale $s$ obtained from Eq. (28) the step $\Delta\omega$ in frequency domain and $\Delta u$ in time domain is calculated according to the formula:

$$\Delta\omega = \frac{1}{s}\sqrt{-8\pi \log(1-\epsilon^2)} \tag{29}$$

$$\Delta u = s\sqrt{-\frac{2}{\pi}\log(1-\epsilon^2)}\,. \tag{30}$$

## Multivariate matching pursuit (MMP)

Need to analyze jointly more than one epoch usually arises in case of simultaneous measurements of more than one signal, or when subsequent epochs are treated as realizations of the same process (usually via some kind of ergodicity assumption). In EEG/MEG analysis these two cases usually refer to:

1. Simultaneous recording of EEG/MEG from more than one electrode/sensor, called hereafter "channels".
2. EEG/MEG recording of subsequent time-locked responses to repetitive stimuli (event-related potentials or fields, ERPs/ERFs), called hereafter "trials".

Several versions of the MMP algorithm were separately proposed for particular applications—we will briefly discuss this issue in section *Discussion*. In this paper we propose a general framework, where variants of the MMP are identified on the basis of mathematical formulation of the two crucial elements of the multivariate algorithm:

A. Inter-channel/inter-trial constrains, that is parameters that we keep constant or allow to change across the channels/trials.
B. Criteria for selection of the function from the dictionary in each MMP iteration—in MMP, contrary to MP, this function is fitted to many epochs at the same time, and optimality of fit to many signals at once can be expressed in different ways.

Such an universal approach allows for a direct implementation of the same code to a multitude of different paradigms encountered in recordings of any multivariate time series, not only EEG/MEG. Nevertheless, through this paper we will stick to "channels" and "epochs" in relation to the organization of the multivariate datasets. The following chapters define basic variants of MMP.

### MMP1 (max sum of the moduli of products, constant phase)

The most straightforward multivariate extension of the MP—let's call it MMP1—can be achieved as follows:

A1. Only the amplitude varies across channels.
B1. We maximize the sum of products in all the channels.

We maximize the sum of moduli of the products rather than their squares, as proposed in [48], due to the more efficient selection of the common phase for all the channels. Also, owing to the Gabor update formula (38), in all the iterations but the first one we compute the products of dictionary's atoms with the function from the previous iteration, which has all the parameters fixed except for the amplitude. This saves a lot of computations comparing to the case of computing products with all the channels residua separately, as in MMP3.

Let us denote the multichannel signal as $\mathbf{x}$, and the signal in the $i$th channel as $\mathbf{x}^i$, with $i = 1 \dots N_c$, where $N_c$ is the number of channels. We may express the condition (B1) for the choice of atom $g_\gamma$ in $n$-th iteration as

$$\max_{g_\gamma \in D} \sum_{i=1}^{N_c} |\langle R^n \mathbf{x}^i, g_\gamma \rangle| \tag{31}$$

The whole procedure can be described as

$$\begin{cases} R^0\mathbf{x} = \mathbf{x} \\ R^n\mathbf{x}^i = \langle R^n\mathbf{x}^i, g_{\gamma_n}\rangle g_{\gamma_n} + R^{n+1}\mathbf{x}^i \\ g_{\gamma_n} = \arg\max_{g_\gamma \in D} \sum_{i=1}^{N_c} |\langle R^n\mathbf{x}^i, g_\gamma\rangle| \end{cases} \tag{32}$$

Results of MMP1 are given in terms of functions $g_{\gamma_n}$, selected in consecutive iterations, and their weights in all the channels, determined for channel $i$ by the real-valued products $\langle R^n\mathbf{x}^i, g_{\gamma_n}\rangle$. In each iteration, the multichannel residuum $R^{n+1}\mathbf{x}$ is computed by subtracting from the previous residua in each channel $i$ the contribution of $g_{\gamma_n}$, weighted by $\langle R^n\mathbf{x}^i, g_{\gamma_n}\rangle$.

#### MMP2 (max modulus of the sum of products, constant phase)

Assumption of invariant phase in all the channels was explored in [22] to yield an efficient decomposition algorithm. If we modify the criterion of choice from the previous section to

$$\max_{g_\gamma \in D} \left| \sum_{i=1}^{N_c} \langle R^n\mathbf{x}^i, g_\gamma\rangle \right|, \tag{33}$$

we get the conditions:

A2. Only the amplitude varies across the channels.
B2. We maximize the absolute value of the sum of products across channels.

Due to the linearity of the residuum operator $R$ [22], this choice allows for implementing a simple trick. Instead of finding in each step the product of each dictionary's waveform with all the channels separately, and then computing their sum (33), in each step we decompose the average signal $\bar{\mathbf{x}}$

$$\bar{\mathbf{x}} = \frac{1}{N_c}\sum_{i=1}^{N_c} \mathbf{x}^i \tag{34}$$

$$\begin{cases} R^0\bar{\mathbf{x}} = \bar{\mathbf{x}} \\ R^n\bar{\mathbf{x}} = \langle R^n\bar{\mathbf{x}}, g_{\gamma_n}\rangle g_{\gamma_n} + R^{n+1}\bar{\mathbf{x}} \\ \mathbf{g}_{\gamma_n} = \arg\max_{\mathbf{g}_\gamma \in D} |\langle R^n\bar{\mathbf{x}}, g_\gamma\rangle| \\ R^n\mathbf{x}^i = \langle R^n\mathbf{x}^i, g_{\gamma_n}\rangle g_{\gamma_n} + R^{n+1}\mathbf{x}^i \end{cases} \tag{35}$$

This procedure yields a computational complexity close to the monochannel MP—compared to MMP1, reduced by the factor $N_c$ (that is, number of channels).

Convergence of this procedure may be relatively slower for waveforms appearing in different channels with exactly opposite phases.

Due to operating on the average of channels, this version of the algorithm cannot be directly applied to the data presented in the average reference (montage). These problems are absent in MMP1 as well as in the next implementation, allowing for arbitrary phases across the channels.

#### MMP3 (variable phase)

A3. Phase and amplitude vary across the channels.
B3. We maximize the sum of squared products (energies) across channels.

Again, as in (31), we maximize

$$\max_{g_\gamma \in D} \sum_{i=1}^{N_c} |\langle R^n \mathbf{x}^i, g^i_\gamma \rangle|^2 \tag{36}$$

but this time $g^i_{\gamma_n}$ are not the same $g_{\gamma_n}$ for all channels $i$—they can have different phases.

$$\begin{cases} R^0 \mathbf{x} = \mathbf{x} \\ R^n \mathbf{x}^i = \langle R^n \mathbf{x}^i, g^i_{\gamma_n} \rangle g^i_{\gamma_n} + R^{n+1} \mathbf{x}^i \\ g_{\gamma_n} = \arg\max_{\mathbf{g}_\gamma \in D} \sum_{i=1}^{N_c} |\langle R^n \mathbf{x}^i, g^i_\gamma \rangle|^2 \end{cases} \tag{37}$$

As presented in the Appendix (section *Optimal phase of a Gabor function*), computing an optimal phase of Gabor function $g_\gamma$, maximizing absolute value of the product $\langle R^n \mathbf{x}^i, g_\gamma \rangle$, can be implemented very efficiently. Value of (31) for phases optimized separately will never be smaller than in the case of the phase common to all the channels, so this freedom should improve the convergence.

#### MMPXY

In case of multichannel recordings of event-related potentials (ERPs), we are dealing with a slightly more complicated structure of the data. Instead of a vector of signals $\mathbf{x}^i$, where $i$ indexes channels/sensors, we get a matrix of signals $\mathbf{X}^i_k$, with additional index $k$ reflecting subsequent repetitions (single trials). On such data, MMP can be performed across either of the two dimensions $i$ and $k$. For the sake of simplicity, we will call these indices "channels" and "trials", although for the algorithm they can represent any direction of a multidimensional dataset. The algorithm does not contain any problem-specific optimizations and as such preserves generality.

Usually, the first index in multiplexed multichannel data is the channel (sensor) $i$ and the second is the time index $t$. Number of trial $k$ comes next, and indexes the the set of whole epochs and all channels $i$, related to the $k$-th repetitions of an event (usually ERP). MP operates on epochs indexed by discrete time $t$. MMP will operate on channel and trial indices $i$ and $k$, allowing for different constrains across sensors or repetitions.

For the mp5 package we assumed the naming convention in the form MMPXY, where X denotes the version of MMP algorithm used in each iteration across the channel dimension, and Y—across the repetitions:

**MMP11:** For each channel and trial fit the optimal $g_\gamma$ with $\phi = \text{const}$.

**MMP12:** Average trials $k$ in each channel $i$ and fit optimal $g_\gamma$ with $\phi = \text{const}$ to these averages.

**MMP21:** For each trial $k$ find the average across channels $i$ and fit optimal $g_\gamma$ with $\phi = \text{const}$ to these averages.

**MMP23:** For each trial $k$ find the average across channels $i$ and fit to these averages optimal $g_\gamma$ with potentially different phase $\phi$ in each channel.

**MMP32:** For each channel $i$ find the average across trials $k$ and fit to these averages optimal $g_\gamma$ with potentially different phase $\phi$ in each trial.

**MMP33:** For each channel and trial fit the optimal $g_\gamma$ with potentially different phase $\phi$ in each channel and trial.

For example, MMP12 denotes a setting where in each iteration the choice of the atom fitting best all the channels will be effectuated according to MMP1, while across the trials

MMP2 will be used. That is, in each iteration the residua of single trials are averaged separately in each channel, and to these averages the best $g_\gamma$ is fitted according to MMP1. Naming of the algorithms (MMPX) corresponds to the three above subsections. MMP13 and MMP31 were not implemented in mp5.

## Discussion

### Resolution of MP

One of the problems, faced by everyone applying MP to exploratory analysis of signals, is "how big should be the dictionary so that I do not miss some important structures?" Of course "the bigger the better", but increasing the size of the dictionary increases significantly the computational cost, and the exact gain in resolution of MP representation was not known so far. Optimal Gabor dictionaries, introduced in this paper, for the first time allow to relate the density of the dictionary to the maximum error of a single iteration of the algorithm.

### Details of implementation

This paper describes mathematical foundations and numerical optimizations, implemented in the freely available software package mp5, developed at the University of Warsaw for decomposition of signals using multivariate matching pursuit. This software made possible several published works [24,28,29] plus some studies in progress. It builds on another decade of experience in using the previous, monovariate version of the algorithm which we made freely available as mp4 at http://eeg.pl/mp, and used in over a dozen published studies. Although all these cited works would not be possible without this software, they were published as "classical scientific papers" where author is supposed to concentrate only on the scientific novelty, not tools. There was no space for all the important details of the implementations, which make a big difference in the results and hence constitute the core of the Reproducible Research. Therefore, although the source code of both these packages have been freely available for years, a complete description of employed mathematical and numerical optimizations and tricks, as presented in Appendix Implementation and optimizations, was not published up to now.

### Applications of MMP to EEG/MEG

Probably the most promising field of future applications is related to the multivariate matching pursuit (MMP). Real world applications were made possible owing to the progress in computer hardware in recent decades. In section *Multivariate matching pursuit (MMP)* we propose, in concordance with [44], a simple classification of MMP variants according to the combinations of the constraints on parameters and criteria of choice. However, there is a potential continuum of other approaches outside of this simple scheme, tailored very specifically for particular applications. Most of them suffer from the need of arbitrary setting some parameter that may significantly change the results of decomposition. In this light we believe that in most cases the simple approach proposed in section *Multivariate matching pursuit (MMP)*, which is free from arbitrary parameters, is optimal and most elegant, at least for starting. A procedure that is free of task-specific settings has also obvious advantages stemming directly from its generality.

However, specific approaches of course may offer addition advantages in particular cases. For example, MMP tailored for the analysis of stereo recordings of sound

in [49] allows for different time positions of the time-frequency atoms present in the two channels. Together with different amplitudes in each channel, it relates to modeling the microphones as gain-delay filters in the anechoic case. Unfortunately, a model explaining relations between channels of EEG/MEG recordings is far more complicated, even in the case of known distribution of sources (so-called forward EEG problem).

An attempt to incorporate constraints, reflecting the generation of multichannel EEG, into the MMP procedure, was presented in [26]. To the purely energetic criterion of MMP1 (31), a second term was added to favor those $g_\gamma$ which give smooth distribution of amplitudes across the channels. Spatial smoothness (quantified by Laplacian operators) means basically that the values of $\langle R^n \mathbf{x}^i, g_\gamma \rangle$ should be similar for $i$ corresponding to the neighboring channels. However, a choice combining two completely different criteria requires some setting of their relative weights. For example, if we attribute too much importance to the spatial criterion, in favor of the energetic one, we may obtain atoms giving very smooth scalp distributions across electrodes. But in such a case the convergence of the MMP procedure, measured in the rate-distortion sense, relating to the amount of explained energy, may be severely impaired. Up to now, no objective or optimal settings for regulating the influence of such extra criteria on the MMP algorithms was proposed.

Finally, we should also point out some of the possible novel and promptly implementable applications of MMP. Variants of the multivariate algorithms, described in section *Multivariate matching pursuit (MMP)*, can be related to several models of multichannel recordings of repetitive trials of evoked brains activity. Algorithms developed for parameterization of EEG structures in subsequent channels [22,24] have been be applied to decomposing subsequent trials of event-related potentials [28,29].

Ongoing works include application of MMP3 to evoked potentials, where variable phase accounts for the variable latencies, and instantaneous decomposition of both repetitions and channels of event-related potentials, with some of the discussed constraints applied separately to the relevant dimensions.

Apart from that, MMP3 may be also used to compute estimates of the phase locking factor [50] (also called inter-trial coherence, [51]). Simultaneous decomposition of all the repetitions will be crucial in this case: in separate MP decompositions of subsequent trials, atoms representing possibly the same structures can have slightly different frequencies, which makes their relative phase insignificant. By estimating the phase coherence only in those are of the time-frequency plane, where indeed an oscillatory activity appears, we may get rid of a lot of noise blurring previously applied estimates.

## Software availability and requirements

Software package described in this article is freely available from http://braintech.pl/svarog/. It can be run on a computer with a reasonably recent version of MS Windows, Mac OS or GNU/Linux with Java runtime environment. Screencasts (video files), showing (1) downloading and configuration of the package and (2) MP decomposition process of a sample EEG epoch via Svarog, are included as [Additional file 1] and [Additional file 2]. These videos can be also viewed in a variety of formats embedded in HTML5 at http://braintech.pl/svarog. [Additional file 3] contains a snapshot of Svarog's help related to mp5.

Complete source code for the MMP engine written in C is available from http://git.braintech.pl/matching-pursuit.git. GUI is implemented in Java within the Svarog system,

for which the source code is available at http://git.braintech.pl/svarog.git. Both projects are released on terms of the General Public License (GPL).

Svarog is a loose acronym for Signal Viewer, Analyzer and Recorder on GPL, and constitutes the core of the world's first professional EEG recording and analysis system based entirely on GPL software (http://braintech.pl/Manifesto.html). This multiplatform system is developped primarily for GNU/Linux. Current versions of the system, including the above discussed software plus the OpenBCI framework for brain-computer interfaces and a modified version of the PsychoPy framework for experiments design, are available as packages for Ubuntu and Debian (see http://deb.braintech.pl).

## Appendix

### Implementation and optimizations

#### *Product update formula*

This optimization was proposed in [39].

Let us recall from (1) the formula for the $n$th residuum:

$$R^n x = \langle R^n x, g_{\gamma_n} \rangle g_{\gamma_n} + R^{n+1} x$$

Taking the product of both sides with $g_{\gamma_i}$, which is the candidate for selection in the next iteration, we get

$$\langle R^{n+1} x, g_{\gamma_i} \rangle = \langle R^n x, g_{\gamma_i} \rangle - \langle R^n x, g_{\gamma_n} \rangle \langle g_{\gamma_n}, g_{\gamma_i} \rangle \tag{38}$$

This equation expresses the product of a dictionary function $g_{\gamma_i}$ with the residuum in step $n+1$ using two products, which were already calculated in the previous iteration—$\langle R^n x, g_{\gamma_i} \rangle$ and $\langle R^n x, g_{\gamma_n} \rangle$—and a product of two functions from the dictionary—$\langle g_{\gamma_n}, g_{\gamma_i} \rangle$. Therefore, the only thing that remains to be computed is a product of two known functions.

Inner product of continuous Gabor functions can be expressed in terms of elementary functions (see [47,52]). Unfortunately, it does not reflect with enough accuracy the numerical value of the product of two discrete vectors, representing sampled versions of the same Gabor functions. Exact formula for the product of the latter involves theta functions, which can be approximated by relatively fast converging series [47].

#### *Sin, Cos, and Exp: fast calculations and tables*

In spite of the trick from the previous section, still—at least in the first iteration—we need to compute the "plain" inner products of the signal with Gabor functions. Using the result from section *Optimal phase of a Gabor function*, of all the phases $\phi$ we calculate products only for $\phi = 0$ and $\phi = \frac{\pi}{2}$.

Computationally, the most expensive part is not the actual calculation of the inner products, but the generation of discrete vectors of samples from Equation (3), which contains cosine and exponent. Compilers usually approximate these functions by high-order polynomials. Although contemporary CPUs may implement directly some special functions, they will still be much more expensive to compute than basic additions or multiplications. Therefore, avoiding explicit calls of these functions may result in significant acceleration—together with tabularization, it accelerated the MP implementation [45] by over an order of magnitude. In the following we show (after [53]) how to fill in a vector with values of sines and cosines for equally spaced arguments using only one call of these functions.

Since the time $t$ in (3) in the actual computations is discrete, the trick is to compute $\sin(\omega(t+1))$ knowing $\sin(\omega t)$. Using the trigonometric identity (59) with its corresponding form for the sine function, we get

$$\sin(\omega(t+1)) = \sin(\omega t + \omega) = \cos(\omega t)\sin(\omega) + \sin(\omega t)\cos(\omega) \tag{39}$$

$$\cos(\omega(t+1)) = \cos(\omega t + \omega) = \cos(\omega t)\cos(\omega) - \sin(\omega t)\sin(\omega) \tag{40}$$

We start with $t = 0$, setting $\cos(0) = 1$ and $\sin(0) = 0$, and computing constants $\cos(\omega)$ and $\sin(\omega)$. Values of (39) and (40) for subsequent $t$ can be filled in a recursive way, using the computed $\cos(\omega)$ and $\sin(\omega)$ and taking as $\sin(\omega t)$ and $\cos(\omega t)$ values from the previous steps.

A similar approach can accelerate computation of the factors $e^{-\alpha t^2}$ present in (3):

$$e^{-\alpha(t+1)^2} = e^{-\alpha t^2 - 2\alpha t - \alpha} = e^{-\alpha t^2} e^{-2\alpha t} e^{-\alpha} \tag{41}$$

To compute (41) we need $e^{-\alpha t^2}$ from the previous iteration, constant $e^{-\alpha}$ independent of $t$, and $e^{-2\alpha t}$. The last factor can be updated in each iteration at a cost of one multiplication: to get $e^{-2\alpha(t+1)}$ from $e^{-2\alpha t}$ we multiply it by a precomputed constant $e^{-2\alpha}$.

In all these cases we also take into account the symmetries $\sin(-x) = -\sin(x)$, $\cos(-x) = \cos(x)$, and $e^{-(-x)^2} = e^{-x^2}$ to double the savings. Values of these vectors can be stored in memory for subsequent calculations of Gabor vectors (3) with different combinations of sin/cos and exp, but only if we restrict the discretization of parameters to some integer grid, for example:

$$u = 1 \ldots N, \quad \omega = (1 \ldots N)\pi/N, \quad s = 1 \ldots N \tag{42}$$

Apart from fast Sin, Cos and Exp function generation, the optimal dictionary allows for saving in the computer memory the tables with values of these functions. The number $N_s$ of different scales in an optimal dictionary is (see Equation (27)):

$$N_s = \lfloor \log_a N \rfloor \tag{43}$$

where $N$ is the number of samples in signal and $a$ is the parameter expressed by Equation (27). A typical epoch of EEG/MEG contains some thousands samples, so it is possible to store all EXP functions in computer memory. Due to the fact, that Gabors, for given scale, are arranged in frequency domain in increments of $\Delta\omega$ (29) one can save also in computer's memory one period of Sine/Cosine signal of the lowest frequency. The sine and cosine signal of higher frequencies, for example $k \times \Delta\omega$, where $k$ is the natural number, can be generated by means of selecting every $k$-th sample from Cos/Sine signal of frequency $\Delta\omega$ stored in the computer's memory.

#### *Fast detection of two orthogonal Gabors*

Update formula (section *Product update formula*) in combination with optimal dictionary allows for uses the next numerical optimization—fast assessment of orthogonality of two Gabors functions. Let us analyze the analytical formula for an inner product of two

Gabors. After substituting to the Equation (18) the exact expression for normalization factor $K_\gamma$ and constant $C$, and introducing the following factors:

$$\begin{aligned} X &= \Big( \cos\left[(\omega_0+\omega_1)B + (\phi_0+\phi_1) - (\omega_0 u_0 + \omega_1 u_1)\right] e^{-\pi \frac{(\omega_0+\omega_1)^2}{4A}} \\ &\quad + \cos\left[(\omega_0-\omega_1)B + (\phi_0-\phi_1) - (\omega_0 u_0 - \omega_1 u_1)\right] e^{-\pi \frac{(\omega_0-\omega_1)^2}{4A}} \Big) \\ Y &= K_0 K_1 \sqrt{s_1 s_2} = \frac{2^{6/4}}{\sqrt{\left(1+\cos(2\phi_0)e^{-\frac{s_0^2\omega_0^2}{2\pi}}\right)\left(1+\cos(2\phi_1)e^{-\frac{s_1^2\omega_1^2}{2\pi}}\right)}} \\ Z &= \sqrt{\frac{\pi}{4A}}(s_1 s_2)^{-\frac{1}{2}} e^C = \sqrt{\frac{s_0 s_1}{4\left(s_0^2+s_1^2\right)}} e^{-\pi \frac{(u_0-u_1)^2}{s_0^2+s_1^2}} \end{aligned} \tag{44}$$

one can obtain:

$$\langle g_0, g_1 \rangle = X \cdot Y \cdot Z \tag{45}$$

It is straightforward to estimate that value of factor $X$ fulfils the condition

$$|X| < 2 \tag{46}$$

for every possible set of parameters $\{s_0, s_1, \omega_0, \omega_1, \phi_0, \phi_1, u_0, u_1\}$. The maximal value of parameter $Y$ is limited by the lowest values of scale $s$ and frequency $\omega$, which, based on (29), are:

$$\begin{aligned} s &= s_{\min} > 0 \\ \omega &= \Delta\omega = \frac{2\sqrt{\pi}}{s_{\min}} \sqrt{-\log(1-\epsilon^2)} \end{aligned} \tag{47}$$

and phases $\phi_0 = \phi_1 = \pm\frac{\pi}{2}$. Substituting above formulae into factor $Y$ results in following expression:

$$|Y| \le \frac{2^{3/2}}{\sqrt{\left(1+\cos(2\phi_0)\left(1-\epsilon^2\right)^2\right)\left(1+\cos(2\phi_1)\left(1-\epsilon^2\right)^2\right)}} \le \frac{2^{3/2}}{\epsilon^2(2-\epsilon^2)} \approx \frac{\sqrt{2}}{\epsilon^2} \tag{48}$$

Based on this observation, it is possible to introduce a numerical threshold, defining an approximate orthogonality of two Gabor functions. The factor $Z$ depends on the relative position and the width of two Gabors. Atoms which differ mostly in these parameters will give a small factor $Z$. Therefore

$$\frac{2\sqrt{2}}{\epsilon^2} Z < \eta \Longrightarrow |XYZ| < \eta \Longrightarrow \langle g_0, g_1 \rangle \approx 0 \tag{49}$$

where $\eta$ can be set for example at the accuracy of a double precision number ($10^{-16}$). Condition (49) allows for efficient detection of orthogonal atoms in dictionary and replacing their inner product by zero in Equation (38). Moreover, in case of a dictionary with uniform step $\Delta\omega$ at a given scale, it is possible to determine the set of Gabor functionss, characterized by the same position $u$, for which inner product with the Gabor selected in previous iteration will be zero.

***Limiting domain of the product***

When Equation (49) is not fulfilled, the two Gabor functions cannot be treated as orthogonal, and their product has to be determined. Full scalar product of two Gabor functions $g_1$ and $g_2$ can be written as

$$\langle g_1, g_2 \rangle = K_1 K_2 \int_{-\infty}^{+\infty} e^{-A(t-B)^2} e^C \cos(\omega_1(t-u_1)+\phi_1)\cos(\omega_2(t-u_2)+\phi_2)\,dt \quad (50)$$

where A, B and C are defined in Equations (15–17).

To perform numerical integration, one can replace the improper integral above with a definite integral on a sufficiently large interval $[a;b]$, so that

$$\left| \int_{-\infty}^{+\infty} g_1(t)g_2(t)\,dt - \int_a^b g_1(t)g_2(t)\,dt \right| < \eta \quad (51)$$

for given error bound $\eta$. Such interval can be constructed as $[B-\Delta; B+\Delta]$ to fulfil

$$\begin{aligned} \left| \int_{B+\Delta}^{+\infty} g_1(t)g_2(t)\,dt \right| &< \frac{1}{2}\eta \\ \left| \int_{-\infty}^{B-\Delta} g_1(t)g_2(t)\,dt \right| &< \frac{1}{2}\eta\,. \end{aligned} \quad (52)$$

Therefore, $\Delta$ can be calculated as

$$\Delta = \frac{1}{\sqrt{A}} \operatorname{erf}^{-1}\left(1 - \frac{\eta\epsilon^2}{\sqrt{2}} e^{-C} \sqrt{\frac{s_1}{s_2} + \frac{s_2}{s_1}}\right), \quad (53)$$

where $\operatorname{erf}^{-1}$ is an inverse of the error function

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt\,. \quad (54)$$

The value of $\Delta$ in (53) is well-defined unless inequality (49) is fulfilled, in which case the atoms are orthogonal to the given precision and there is no need to define integration interval.

To simplify formula (53), one can use inequality

$$\operatorname{erf}^{-1}(1-x) \le \sqrt{-\log x}\,, \quad (55)$$

which is fulfilled for all $x \in (0;1]$. Therefore,

$$\Delta' = \frac{1}{\sqrt{A}} \sqrt{C - \log\left(\frac{\eta\epsilon^2}{\sqrt{2}} \sqrt{\frac{s_1}{s_2} + \frac{s_2}{s_1}}\right)} \quad (56)$$

is guaranteed to fulfil $\Delta' \ge \Delta$.

***Optimal phase of a Gabor function***

In the following we find an explicit formula for the phase $\phi_{\max}$, that maximizes the product of signal $x$ with Gabor function of given time position $u$, frequency $\omega$, and scale $s$. Let us recall the formula (3) of a real Gabor function:

$$g_\gamma(t) = K(\gamma) e^{-\pi\left(\frac{t-u}{s}\right)^2} \cos\left(\omega(t-u)+\phi\right).$$

$\gamma$ denotes the set of parameters $\gamma = \{u, s, \omega, \phi\}$ and $K(\gamma)$ is such that $||g_\gamma|| = 1$. Writing $K(\gamma)$ explicitly gives

$$g_{(\gamma)}(t) = \frac{e^{-\pi\left(\frac{t-u}{s}\right)^2} \cos\left(\omega(t-u)+\phi\right)}{||e^{-\pi\left(\frac{t-u}{s}\right)^2} \cos\left(\omega(t-u)+\phi\right)||} \quad (57)$$

Phase shift $\phi$ can be also expressed as a superposition of two orthogonal oscillations $\cos(\omega(t-u)+\phi)$ and $\sin(\omega(t-u)+\phi)$. We define

$$\begin{aligned} C &= e^{-\pi\left(\frac{t-u}{s}\right)^2}\cos(\omega(t-u)) \\ S &= e^{-\pi\left(\frac{t-u}{s}\right)^2}\sin(\omega(t-u)) \end{aligned} \tag{58}$$

and, using the trigonometric identity

$$\cos(\alpha+\phi) = \cos\alpha\cos\phi - \sin\alpha\sin\phi, \tag{59}$$

we write the Gabor function (57) as

$$g_\gamma(t) = \frac{C\cos\phi - S\sin\phi}{\|C\cos\phi - S\sin\phi\|} \tag{60}$$

Using $\|x\|^2 = \langle x, x\rangle$, and the orthogonality of $C$ and $S$ defined in (58) ($\langle C, S\rangle = 0$), we write the product of Gabor function defined as (60) with the signal $x$ as

$$\langle x, g_\gamma\rangle = \frac{\langle x, C\rangle\cos\phi - \langle x, S\rangle\sin\phi}{\|C\cos\phi - S\sin\phi\|} = \frac{\langle x, C\rangle - \langle x, S\rangle\tan\phi}{\sqrt{\langle C, C\rangle + \langle S, S\rangle\tan^2\phi}} \tag{61}$$

We are looking for the maximum absolute value of this product. For the sake of simplicity we will maximize $\langle x, g_\gamma\rangle^2$ instead of $|\langle x, g_\gamma\rangle|$. Denoting $\nu = \tan\phi$

$$\langle x, g_\gamma\rangle^2 = \frac{(\langle x, C\rangle - \langle x, S\rangle\nu)^2}{\langle C, C\rangle + \langle S, S\rangle\nu^2} \tag{62}$$

To find $\nu$ which maximizes (62), we look for zeros of the derivative $\frac{\partial}{\partial\nu}\langle x, g_\gamma\rangle^2$ and find two roots:

$$\nu_1 = \frac{\langle x, C\rangle}{\langle x, S\rangle}, \quad \nu_2 = -\frac{\langle x, S\rangle\langle C, C\rangle}{\langle x, C\rangle\langle S, S\rangle}$$

Substituting these values for $\tan\phi$ in (61) we get

$$\langle x, g_\gamma\rangle|_{\nu=\nu_1} = 0$$

$$\langle x, g_\gamma\rangle|_{\nu=\nu_2} = \frac{\langle x, C\rangle + \frac{\langle x,S\rangle^2\langle C,C\rangle}{\langle x,C\rangle\langle S,S\rangle}}{\sqrt{\langle C, C\rangle + \frac{\langle x,S\rangle^2\langle C,C\rangle^2}{\langle x,C\rangle^2\langle S,S\rangle}}}$$

Since for $\nu_1$ the square of the product is minimum (zero), the other extremum is a maximum in $\nu_2$. Therefore, the phase $\phi$ that maximizes $\langle x, g_\gamma\rangle^2$ is given by

$$\phi_{\max} = \arctan\frac{\langle x, S\rangle/\langle S, S\rangle}{\langle x, C\rangle/\langle C, C\rangle} \tag{63}$$

and the maximum value of the product is

$$\langle x, g_\gamma\rangle_{\max} = \frac{\langle x, C\rangle^2/\langle C, C\rangle + \langle x, S\rangle^2/\langle S, S\rangle}{\sqrt{\langle x, C\rangle^2/\langle C, C\rangle + \langle x, S\rangle^2/\langle S, S\rangle}} \tag{64}$$

#### *Applying Fast Fourier Transform*

Estimation of the product of a Gabor function with signal $x$ (61) requires computing of the inner product of signal $x$ with functions $C$ and $S$ defined in Equation (58). Let us rewrite the formulae for $< x, C >$ and $< x, S >$:

$$\langle x, C\rangle = \int_{-\infty}^{\infty} x e^{-\pi\left(\frac{t-u}{s}\right)^2}\cos(\omega(t-u)) \tag{65a}$$

$$\langle x, S\rangle = \int_{-\infty}^{\infty} x e^{-\pi\left(\frac{t-u}{s}\right)^2}\sin(\omega(t-u)) \tag{65b}$$

Multiplying (65a) by $i = \sqrt{-1}$ and substracting it from (65b), one can obtain the following equation:

$$\langle x, C\rangle - i\langle x, S\rangle = \int_{-\infty}^{\infty} x e^{-\pi\left(\frac{t-u}{s}\right)^2} e^{-i\omega(t-u)} \tag{66}$$

The right side of Equation (66) is the Fourier Transform of signal $x$ windowed by Gauss functions $e^{-\pi\left(\frac{t-u}{s}\right)^2}$. This formula allows for fast computation of inner products $< x, C >$ and $< x, S >$, since in and optimal dictionary the atoms with the given scale $s$ are arranged in frequency domain with uniform step $\Delta\omega$ (see Equation (29)).

#### *Additional structures in the dictionary*

Apart from the Gabor functions, Gabor dictionary implemented in mp5 contains also the following functions:

- "Pure" harmonic waves

  $$S(t) = K_s \cos(\omega t + \phi) \tag{67}$$

  where $K_s$ is normalization factor such that $\langle S(t), S(t)\rangle = 1$ on the analysed signal length. The phase $\phi$ of the signal $S(t)$ is estimated according to Equation (63). Harmonic functions are distributed in frequency domain with step $\Delta\omega$ determined by Equation (29);
- Kronecker delta functions

  $$\Delta(t-u) = \begin{cases} 1, \text{for } t = u \\ 0, \text{for } t \neq u \end{cases} \tag{68}$$

  In this work, the delta functions are distributed across the whole time domain, that is for each point of the time series.
- "pure" Gaussians

  $$G(t) = K_g e^{-\pi\frac{(t-u)^2}{s^2}} \tag{69}$$

  Where $K_g$ is normalization factor such that $\langle G(t), G(t)\rangle = 1$ on the analysed signal length. These functions are distributed in the scale domain with step $a$ (see Equation (27)) and with step $\Delta u$ (30) in the time domain.

## Additional files

**Additional file 1: A video tutorial for downloading and configuration of the Svarog package, used for computations and visualization of results.**

**Additional file 2: Shows the steps from loading the signal into Svarog to displaying interactive map of time-frequency energy distribution.**

**Additional file 3: Contains help of the MP module from Svarog.**

#### Abbreviations

EEG: Electroencephalogram; FFT: Fast Fourier Transform; MEG: Magnetoencephalogram; MP: Matching pursuit; MMP: Multivariate matching pursuit; S/N: Signal to noise; SWA: Slow wave activity.

#### Competing interests

The authors declare that they have no competing interests.

#### Authors' contributions

RK wrote from the scratch the the mp5 implementation in C, and contributed most of the text in Appendix Implementation and optimizations. PTR derived the formulae for product-related metric and resulting dictionary construction, adapted Svarog and its GUI to the modifications in the algorithm, and wrote most of the sections introducing optimal sampling of Gabor dictionaries. PJD proposed the idea of optimal sampling of Gabor dictionaries

based upon the product-related metrics in 2003, over the next decade supervised and coordinated projects which led to this article and the accompanying software, and wrote the remaining text. All authors read and approved the final manuscript.

**Acknowledgements**
This work was supported from Polish funds for science, including grant from the Polish Ministry of Science and Higher Education (Decision 644/N-COST/2010/0). Authors are grateful for constructive remarks of prof. Kenneth Foster on the manuscript organization and style of presentation. Finally, we thank our colleagues Dobiesław Ircha, Marek Barwiński and Artur Matysiak for years of collaboration on related issues and inspiring discussions.

Received: 3 August 2013 Accepted: 2 September 2013
Published: 23 September 2013

**References**

1. Durka PJ, Blinowska KJ: **Analysis of EEG transients by means of matching pursuit.** *Ann Biomed Eng* 1995, **23:**608–611.
2. Durka PJ, Szelenberger W, Blinowska K, Androsiuk W, Myszka M: **Adaptive time-frequency parametrization in pharmaco EEG.** *J Neurosci Methods* 2002, **117:**65–71.
3. Lelic D, Olesen AE, Brock C, Staahl C, Drewes AM: **Advanced pharmaco-EEG reveals morphine induced changes in the brain's pain network.** *J Clin Neurophysiol* 2012, **29**(3):219–225.
4. Koubeissi MZ, Jouny CC, Blakeley JO, Bergey GK: **Analysis of dynamics and propagation of parietal cingulate seizures with secondary mesial temporal involvement.** *Epilepsy Behav* 2009, **14:**108–112. [http://www.sciencedirect.com/science/article/pii/S1525505008002746]
5. Jouny CC, Adamolekun B, Franaszczuk PJ, Bergey GK: **Intrinsic ictal dynamics at the seizure focus: effects of secondary generalization revealed by complexity measures.** *Epilepsia* 2007, **48**(2):297–304. [http://dx.doi.org/10.1111/j.1528-1167.2006.00963.x]
6. Jouny CC, Franaszczuk PJ, Bergey GK: **Characterization of epileptic seizure dynamics using Gabor atom density.** *Clin Neurophysiol* 2003, **114:**426–437.
7. Jouny CC, Franaszczuk PJ, Bergey GK: **Signal complexity and synchrony of epileptic seizures: is there an identifiable preictal period?** *Clinph* 2005, **116:**552–558.
8. Bergey GK, Franaszczuk PJ: **Epileptic seizures are characterized by changing signal complexity.** *Clin Neurophysiol* 2001, **112:**241–249.
9. Wilson SB, Scheuer ML, Emerson RG, Gabor AJ: **Seizure detection: evaluation of the reveal algorithm.** *Clin Neurophysiol* 2004, **115**(10):2280–2291.
10. Zhang ZG, Yang JL, Chan SC, Luk K, Hu Y: **Time-frequency component analysis of somatosensory evoked potentials in rats.** *BioMed Eng OnLine* 2009, **8:**4. [http://www.biomedical-engineering-online.com/content/8/1/4]
11. Zhang Z, Luk KDK, Hu Y: **Identification of detailed time-frequency components in somatosensory evoked potentials.** *Neural Syst Rehabil Eng, IEEE Trans* 2010, **18**(3):245–254.
12. Zhang ZG, Yang JL, Chan SC, Luk K, Hu Y: **Time-frequency component analysis of somatosensory evoked potentials in rats.** *BioMed Eng OnLine* 2009, **8:**4. [http://www.biomedical-engineering-online.com/content/8/1/4]
13. Schönwald S, Carvalho D, de Santa-Helena E, Lemke N, L Gerhardt G: **Topography-specific spindle frequency changes in obstructive sleep apnea.** *BMC Neurosci* 2012, **13:**89. [http://www.biomedcentral.com/1471-2202/13/89]
14. Schönwald SV, Carvalho DZ, Dellagustin G, de Santa-Helena EL, Gerhardt GJ: **Quantifying chirp in sleep spindles.** *J Neurosci Methods* 2011, **197:**158–164. [http://www.sciencedirect.com/science/article/pii/S0165027011000525]
15. Cervenka MC, Franaszczuk PJ, Crone NE, Hong B, Caffo BS, Bhatt P, Lenz FA, Boatman-Reich D: **Reliability of early cortical auditory gamma-band responses.** *Clin Neurophysiology* 2013, **124:**70–82.
16. Ray S, Maunsell JHR: **Different origins of gamma rhythm and high-gamma activity in macaque visual cortex.** *PLoS Biol* 2011, **9**(4):e1000610.
17. Lelic D, Olesen SS, Valeriani M, Drewes AM: **Brain source connectivity reveals the visceral pain network.** *NeuroImage* 2012, **60:**37–46. [http://www.sciencedirect.com/science/article/pii/S1053811911013991]
18. Drewes AM, Gratkowski M, Sami SAK, Dimcevski G, Funch-Jensen P, Arendt-Nielsen L: **Is the pain in chronic pancreatitis of neuropathic origin? Support from EEG studies during experimental pain.** *World J Gastroenterol* 2008, **14**(25):4020–4027. [http://www.biomedsearch.com/nih/pain-in-chronic-pancreatitis-neuropathic/18609686.html]
19. Żygierewicz J, Kelly EF, Blinowska KJ, Durka PJ, Folger S: **Time-frequency analysis of vibrotactile driving responses by matching pursuit.** *J Neurosci Methods* 1998, **81:**121–129.
20. Durka PJ, Ircha D, Neuper C, Pfurtscheller G: **Time-frequency microstructure of event-related desynchronization and synchronization.** *Med Biol Eng Comput* 2001, **39**(3):315–321.
21. Durka PJ: **Time-frequency microstructure and statistical significance of ERD and ERS.** In *Progress in Brain Research. Volume 159*. Edited by Neuper C, Klimesch W: Elsevier BV; 2006:121–133.
22. Durka PJ, Matysiak A, Montes EM, Sosa PV, Blinowska KJ: **Multichannel matching pursuit and EEG inverse solutions.** *J Neurosci Methods* 2005, **148:**49–59.
23. Lelic D, Gratkowski M, Valeriani M, Arendt-Nielsen L, Drewes AM: **Inverse modeling on decomposed electroencephalographic data: a way forward?** *J Clin Neurophysiol* 2009, **26**(4):227–235. [http://www.biomedsearch.com/nih/Inverse-modeling-decomposed-electroencephalographic-data/19584750.html]
24. Zwoliński P, Roszkowski M, Żygierewicz J, Haufe S, Nolte G, Durka P: **Open database of epileptic EEG with MRI and postoperational assessment of foci—real world verification for the EEG inverse solutions.** *Neuroinformatics* 2010, **8:**285–299.

25. Bénar C, Papadopoulo T, Clerc M: **Topography time-frequency atomic decomposition for event related M/EEG signals.** In *Proceedings of 29th Annual International IEEE EMBS Conference*; 2007:5461–5464. [ftp://ftp-sop.inria.fr/odyssee/Publications/2007/benar-papadopoulo-etal:07.pdf]
26. Studer D, Hoffmann U, Koenig T: **From EEG dependency multichannel matching pursuit to sparse topographic decomposition.** *J Neurosci Methods* 2006, **153**(2):261–275.
27. Xu P, Yao D: **A novel method based on realistic head model for EEG denoising.** *Comput Methods Programs Biomed* 2006, **83**(2):104–110.
28. Sielużycki C, Kus R, Matysiak A, Durka P, Koenig R: **Multivariate matching pursuit in the analysis of single-trial latency of the auditory M100 acquired with MEG.** *Int J Bioelectromagnetism* 2009, **11**(4):155–160.
29. Sielużycki C, König R, Matysiak A, Kuś R, Ircha D, Durka P: **Single-trial evoked brain responses modeled by multivariate matching pursuit.** *IEEE Trans Biomed Eng* 2009, **56:**74–82.
30. Bénar C, Papadopoulo T, Torrésani B, Clerc M: **Consensus matching pursuit for multi-trial EEG signals.** *J Neurosci Methods* 2009, **180:**161–170. [http://www.sciencedirect.com/science/article/B6T04-4VWHVX5-2/2/e6ebdc581a60cde843503fe30f9940d1]
31. Jörn M, Sielużycki C, Matysiak M, Żygierewicz J, Scheich H, Durka P, König R: **Single-trial reconstruction of auditory evoked magnetic fields by means of template matching pursuit.** *J Neurosci Methods* 2011, **199:**119–128. [http://www.sciencedirect.com/science/article/pii/S0165027011002238]
32. Durka PJ: **On the methodological unification in electroencephalography.** *BioMed Eng OnLine* 2005, **4**(15).
33. Żygierewicz J, Blinowska KJ, Durka PJ, Szelenberger W, Niemcewicz S, Androsiuk W: **High resolution study of sleep spindles.** *Clin Neurophysiol* 1999, **110**(12):2136–2147.
34. Durka PJ, Malinowska U, Szelenberger W, Wakarow A, Blinowska KJ: **High resolution parametric description of slow wave sleep.** *J Neurosci Methods* 2005, **147:**15–21.
35. Durka PJ: **Adaptive time-frequency parametrization of epileptic EEG spikes.** *Phys Rev E* 2004, **69**(051914). [http://pre.aps.org/abstract/PRE/v69/i5/e051914]
36. Nuwer M: **Assesment of digital EEG, quantitative EEG, and EEG brain mapping: report of the American Academy of Neurology and the American Clinical Neurophysiology Society.** *Neurology* 1997, **49:**277–292.
37. Rechtschaffen A, Kales A(Eds): *A Manual of Standardized Terminology, Techniques and Scoring System for Sleep Stages in Human Subjects. No. 204 in National Institutes of Health Publications*. Washington DC: US Government Printing Office; 1968.
38. Iber C, Ancoli-Israel S, Chesson A, Quan S: *The AASM Manual for the Scoring of Sleep and Associated Events: Rules, Terminology and Technical Specification.* American Academy of Sleep Medicine; 2007.
39. Mallat S, Zhang Z: **Matching Pursuit with time-frequency dictionaries.** *IEEE Trans Signal Process* 1993, **41:**3397–3415.
40. Malinowska U, Klekowicz H, Wakarow A, Niemcewicz S, Durka P: **Fully parametric sleep staging compatible with the classical criteria.** *Neuroinformatics* 2009, **7**(4):245–253.
41. Schonwald S, Desantahelena E, Rossatto R, Chaves M, Gerhardt G: **Benchmarking matching pursuit to find sleep spindles.** *J Neurosci Methods* 2006, **156**(1–2):314–321. [http://dx.doi.org/10.1016/j.jneumeth.2006.01.026]
42. Durka PJ, Ircha D, Blinowska KJ: **Stochastic time-frequency dictionaries for matching pursuit.** *IEEE Trans Signal Process* 2001, **49**(3):507–510.
43. Vleeschouwer CD, Zakhor A: **In-loop atom modulus quantization for matching pursuit and its application to video coding.** *IEEE Trans Image Process* 2003, **12**(10):1226–1242.
44. Durka PJ: *Matching Pursuit and Unification in EEG Analysis.* Artech House; 2007. [*Engineering in Medicine and Biology*], [ISBN 978-1-58053-304-1]
45. Ircha D: **MP4—software for matching pursuit with stochastic Gabor dictionaries.** [http://eeg.pl/mp20002009]
46. Tropp JA: **Constructing packings in projective spaces and Grassmannian spaces via alternating projection.** ICES Report 04-23, UT-Austin 2004.
47. Ferrando SE, Doolittle EJ, Bernal AJ, Bernal LJ: **Probabilistic matching pursuit with Gabor dictionaries.** *Signal Process* 2000, **80**(10):2099–2120.
48. Gribonval R: **Piecewise linear source separation.** In *Proc. SPIE 03, Volume 5207 Wavelets: Applications in Signal and Image Processing*. San Diego; 2003. [http://spiedigitallibrary.org/volume.aspx?volumeid=2241]
49. Gribonval R: **Sparse decomposition of stereo signals with Matching Pursuit and application to blind separation of more than two sources from a stereo mixture.** *Acoustics, Speech, Signal Process, Proc ICASSP'02, Orlando, Florida, USA* 2002, **3:**3057–3060.
50. Tallon-Baudry C, Bertrand O, Delpuech C, Pernier J: **Stimulus specificity of phase-locked and non-phase-locked 40 Hz visual responses in human.** *J Neurosci* 1996, **16**(13):4240–4249.
51. Delorme A, Makeig S: **EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis.** *J Neurosci Methods* 2004, **134:**9–21.
52. Barwiński M: **Product-based metric for Gabor functions and its implications for the matching pursuit algorithm.** *Master's thesis*. Warsaw University, Institute of Experimental Physics 2004. http://eeg.pl/Members/mbarwinski/m.sc.-on-matching-pursuit-theory
53. Ircha D: **Reprezentacje sygnałów w redundantnych zbiorach funkcji.** *Master's thesis*. University of Warsaw, Faculty of Physics 1997.

doi:10.1186/1475-925X-12-94
**Cite this article as:** Kuś *et al.*: Multivariate matching pursuit in optimal Gabor dictionaries: theory and software with interface for EEG/MEG via Svarog. *BioMedical Engineering OnLine* 2013 **12**:94.

# Artykuł B: *Normalization effects in matching pursuit algorithm with Gabor dictionaries*

Piotr T. Różański (2018). „Normalization effects in matching pursuit algorithm with Gabor dictionaries". W: *Journal of Applied Computer Science* 26.2, s. 187–199. ISSN: 1507-0360. URL: **https://it.p.lodz.pl/file.php/12/2018-2/jacs-2-2018-rozanski.pdf**

JOURNAL OF APPLIED
COMPUTER SCIENCE
Vol. 26 No. 2 (2018), pp. 187-199

# Normalization Effects in Matching Pursuit Algorithm with Gabor Dictionaries

**Piotr T. Różański**

*College of Inter-Faculty Individual Studies*
*in Mathematics and Natural Sciences (MISMaP)*
*University of Warsaw*
*ul. Stefana Banacha 2C, 02-097 Warszawa*
*ptr@mimuw.edu.pl*

**Abstract.** *The matching pursuit (MP) algorithm is a greedy method for signal decomposition used in video coding, data compression, and, particularly, analysis of EEG signals in various paradigms, including P300 and ER(D)S (motor imagery). An important issue for MP implementation is a correct treatment of normalization of atoms (functions) used in computations. Failing to account for normalization-related effects may affect both the numerical stability and the reliability of the algorithm. This paper describes these normalization effects, evaluates their impact on the algorithm's performance, and describe the proper approach together with a ready-to-use implementation, available under a General Public Licence (GPL). Several performance optimizations used as a part of this implementation are also described.*
**Keywords:** *matching pursuit, time-frequency, wavelet, EEG analysis.*

## 1. Introduction

### 1.1. Matching pursuit

The Matching Pursuit (MP) algorithm was first proposed by Mallat and Zhang [1] as a greedy solution to decompose a given signal $x$ into a linear combination of

functions $(g_n)$ from a predefined set $D$, called a dictionary:

$$x = \sum_{n=0}^{M} \alpha_n g_n \,. \tag{1}$$

In general, problem (1) cannot be solved in polynomial time, if functions in the dictionary do not form an orthogonal set. However, a greedy procedure can be defined as

$$\begin{aligned} R_0 x &= x \\ R_{n+1} x &= R_n x - \langle R_n x, g_n \rangle \, g_n \\ g_n &= \operatorname{argmax}_{g_n \in D} \langle R_n x, g_n \rangle \,, \end{aligned} \tag{2}$$

where $x$ is the signal being decomposed, $R_n x$ is a residual before n-th iteration (starting from $n = 0$), and $g_n$ is the atom selected in n-th iteration. The above formulation assumes that all atoms $g \in D$ are $L^2$-normalized, i.e. $\langle g(t), g(t) \rangle = \int g(t)^2 dt = 1$.

In each step, the best approximation to the current residual $R_n x$ is chosen from the "dictionary" $D$ and subtracted from the signal after being multiplied (fitted) with an adaptive scale factor.

It is worth noting, that since the matching pursuit does not provide an exact solution for (1), it should be referred to as "heuristic". However, it is traditionally referred to as an algorithm, and this article will refer to it as such.

The usual choice for the atoms (functions) forming the dictionary is the family of the Gabor atoms $g(s, f_0, t_0, \phi)$ defined as

$$g_{(s,f_0,t_0,\phi)}(t) = K_r \, e^{-\pi\left(\frac{t-t_0}{s}\right)^2} \cos(2\pi f_0 (t - t_0) + \phi) \,,$$

where $K_r$ is the normalization constant.

There are two main reasons for the choice of Gabor atoms. Firstly, the Gabor atoms have the most compact representation on a time-frequency plane, and therefore are a natural candidate for using matching pursuit in calculating high-resolution estimates of the time-frequency distribution of the signal's energy. This approach has been successfully used to describe the time-frequency microstructure of EEG event-related (de-)synchronization in [2].

Secondly, the Gabor atoms and multivariate (multi-channel) matching pursuit have proven to be very useful in solving some advanced problems in EEG analysis,

such as parametrization of the single-trial evoked potentials [3], providing efficient pre-processing for the inverse problem in EEG [4], or parametrization of EEG transients [5].

## 1.2. Dictionary construction

We start by defining a dictionary for the MP algorithm, by adapting the "optimal dictionary" construction from [6]. The construction is based on a single parameter $\epsilon$, related to the density of the dictionary. Smaller values of $\epsilon$ correspond to more fine-grained dictionaries (consisting of a larger number of atoms). Generally, $\epsilon$ should be close to 0 (e.g. 0.1) for accurate decomposition.

Similarly to the original dyadic dictionary by Mallat [1], the scale parameter varies exponentially, starting from a pre-defined minimal scale $s_0$ (which could be problem-specific): $s_0, s_0 a, s_0 a^2, \ldots$ whereas the dilation factor $a$ is defined as

$$a = \frac{1 + \epsilon\sqrt{(2-\epsilon^2)(\epsilon^4 - 2\epsilon^2 + 2)}}{(1-\epsilon^2)^2}\,. \tag{3}$$

For a given scale $s$, frequency ($f_0$) and postion ($t_0$) parameters form a regular, rectangular grid with spacing in both directions defined as

$$\Delta f = \frac{1}{s}\sqrt{-\frac{2}{\pi}\log(1-\epsilon^2)} \tag{4}$$

$$\Delta t = s\sqrt{-\frac{2}{\pi}\log(1-\epsilon^2)}\,. \tag{5}$$

The phase parameter $\phi$ is not taken into account due to the "phase-related equivalence" described in [6] and the reasons which shall be described later in this manuscript.

Such construction of the dictionary guarantees the constraints (in terms of the inner-product-related metric) between adjacent atoms, i.e.

$$\begin{aligned}
\sqrt{1 - \langle g(s, f_0, t_0), g(sa, f_0, t_0)\rangle} &\leq \epsilon \\
\sqrt{1 - \langle g(s, f_0, t_0), g(s, f_0 + \Delta f, t_0)\rangle} &\leq \epsilon \\
\sqrt{1 - \langle g(s, f_0, t_0), g(s, f_0, t_0 + \Delta t)\rangle} &\leq \epsilon\,.
\end{aligned} \tag{6}$$

## 2. Normalization

Since in formula (2) it is assumed that all atoms $g$ are $L^2$-normalized, it is crucial to fulfill this assumption for all atoms in the dictionary. The normalization factor for real Gabor atoms, which can be derived from $\langle g, g\rangle = 1$, is equal to

$$K_r = \frac{2^{3/4}}{\sqrt{s}} \left(1 + \cos(2\phi)\exp(-2\pi s^2 f_0^2)\right)^{-1/2} . \tag{7}$$

This value differs from the normalization value for the envelope function $e^{-\pi(\frac{t}{s})^2}$ by a constant factor of $\sqrt{2}$ and introduces the additional factor involving an atom's phase. We can note that for sufficiently large frequencies ($sf_0 \gg 1$) the normalization constant is asymptotically phase-independent:

$$K_{r\mathrm{HF}} = \lim_{f_0\to\infty} K_r = \frac{2^{3/4}}{\sqrt{s}} . \tag{8}$$

However, in case of discrete-time signals, the above formulae are valid only in the range of low frequencies. Whenever $f_0$ approaches the Nyquist frequency $f_N$, the time discretisation effects come into account and the above formula is no longer accurate.

To derive the correct formula for a fast-oscillating atom with frequency $f_0 = f_N - \delta f$ (with $\delta f \ll f_N$), one can rewrite the oscillating factor of the Gabor atom as

$$\begin{aligned}&\cos(2\pi(f_N - \delta f)(t - t_0) + \phi) = \\ &= (-1)^n \cos(2\pi\delta f(t - t_0) + (2\pi f_N t_0 - \phi)) .\end{aligned}$$

Therefore, for $f \approx f_N$ there is a need to calculate "corrected" phase and frequency

$$f' = f_N - f \tag{9}$$

$$\phi' = 2\pi f_N t_0 - \phi \tag{10}$$

so these values can be used exclusively to calculate the corrected normalization factor

$$K'_r = \frac{2^{3/4}}{\sqrt{s}} \left(1 + \cos(2\phi')\exp(-2\pi s^2 f_0'^2)\right)^{-1/2} . \tag{11}$$

Plot 1 compares the different values of normalization factor: the analytically-derived value of $K_r$, the phase-independent approximation $K_{r\mathrm{HF}}$, and the corrected value of high-frequency normalization factor $K'_r$.

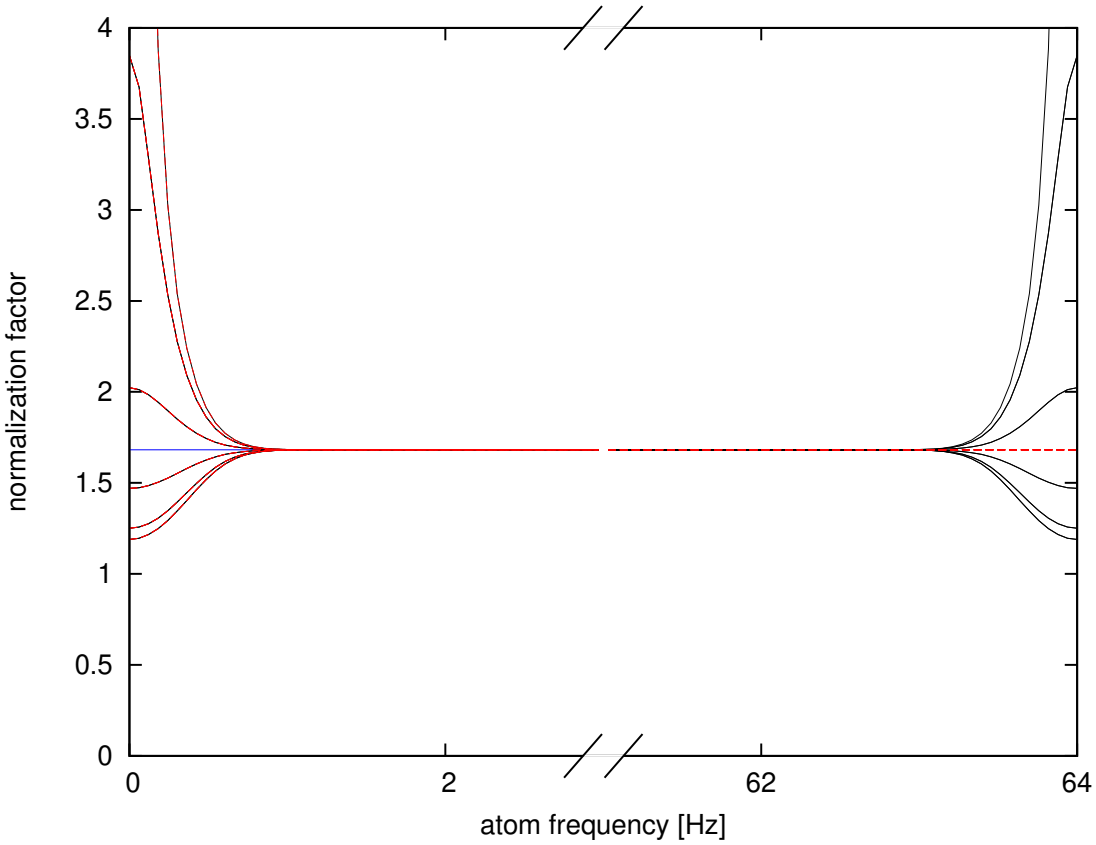


Figure 1: Values of normalization factor $K$ for different normalization strategies (blue—$K_{r\mathrm{HF}}$, red—$K_r$, black—combined $K_r$ and $K'_r$) for $s = 1$ s and sampling frequency of 128 Hz. Multiple branches of the same colour correspond to different values of $\phi$.

Even though the difference between $K_r$ and $K'_r$ is significant only for $f \approx f_N$, atoms of such frequencies appear in real-world signal decompositions on a regular basis. If the normalization is not accounted for correctly, two effects may appear:

- selecting (and substracting) high-frequency atoms will still leave some non-zero residual in the signal, allowing it to be selected as the best match also in subsequent iterations,
- high-frequency atom may be selected instead of the correct one if the error in $K_r$ estimation leads to over-estimating $\langle R_n x, g_n \rangle$ with the current residual.

The proper analytical treatment of normalization would be to use formula $K_r$ for small frequencies ($f < \frac{1}{2} f_N$) and formula $K'_r$ for large frequencies. Errors in normalization factor values obtained this way, relative to the values obtained numerically for the discretely-sampled functions are presented in plot 2. The proposed approximation is quite sufficient even for double-precision calculation, with relative errors not exceeding $10^{-12}$.

## 2.1. Evaluation

As a test, 100 matching pursuit decompositions of randomly-generated white noise signal segments were performed. Each signal segment consisted of 2048 samples with sampling frequency of 128 Hz. Figure 3 visualizes decomposition accuracy, defined as

$$1 - \frac{\text{residual energy}}{\text{signal energy}} = 1 - \frac{\sum_i (R_n x)_i^2}{\sum_i x_i^2} \tag{12}$$

(where $\sum_i$ is a summation over all signal samples) as a function of the number of iterations $n$, both for correct and incorrect (based only on $K_r$) normalization strategies.

Without the correct treatment of atom normalization, all white noise decompositions demonstrated incorrect behaviour. After several initial steps, the algorithm usually started to select the same high-frequency atom in subsequent iterations.

To evaluate this effect on a more realistic example, a similar analysis was performed on eighteen 20-second segments of a single channel EEG recording. The signal was filtered with a high-pass filter to remove DC offset and drift. Results are presented in Figure 4. Although the difference is less pronounced than in the case of white noise decomposition, it is still significant.

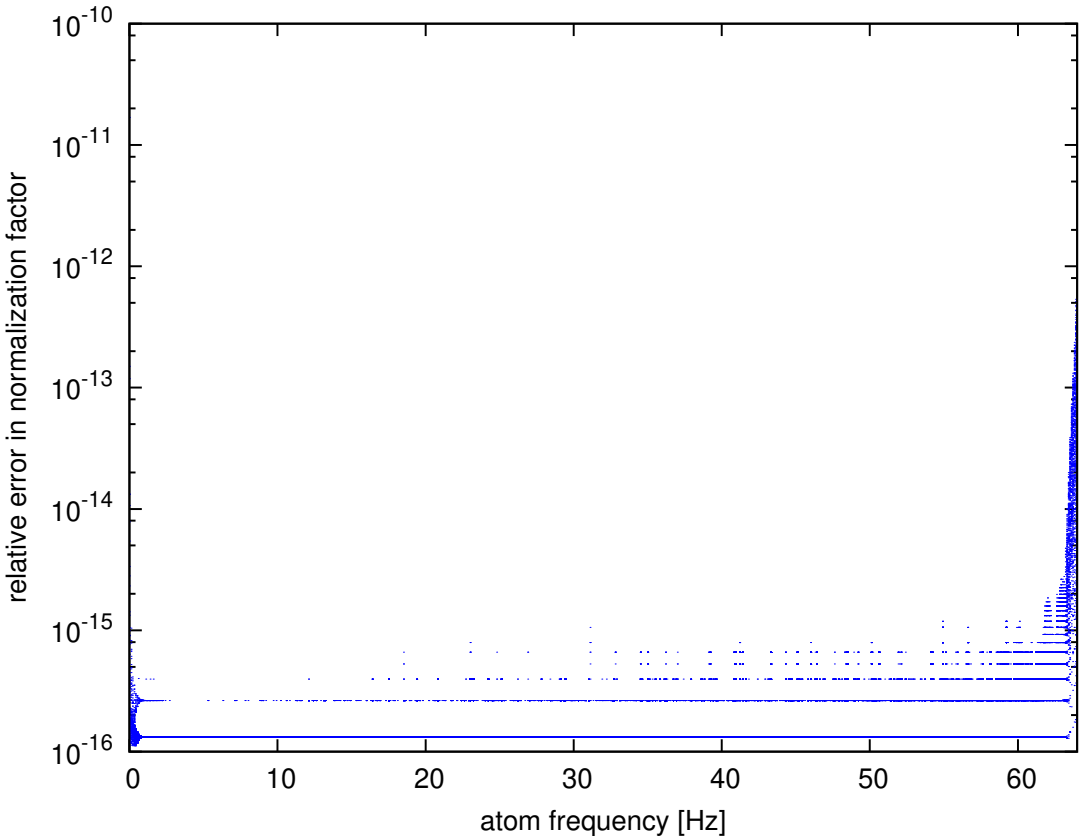


Figure 2: Relative error for the proposed normalization strategy for $s = 1$ s and sampling frequency of 128 Hz.

## 2.2. Discussion

Since the standard approach in biomedical signal analysis is to apply low-pass filtering prior to the MP decomposition, the erroneous behavior is probably unnoticed in most cases, and therefore neglected in many implementations. However, even for the filtered signals, high frequency atoms can still appear because of accumulated numerical residues from previous iterations.

The alternative approach to this problem would be to numerically re-normalize the best atom found in a given iteration to acquire the correct value of $\alpha$ (see eq. 1). However, this may lead to selecting a sub-optimal atom in any given iteration, since un-normalized products $\langle R_n x, g_n \rangle$, used as a criterion for selecting the best atom, could not be properly compared.

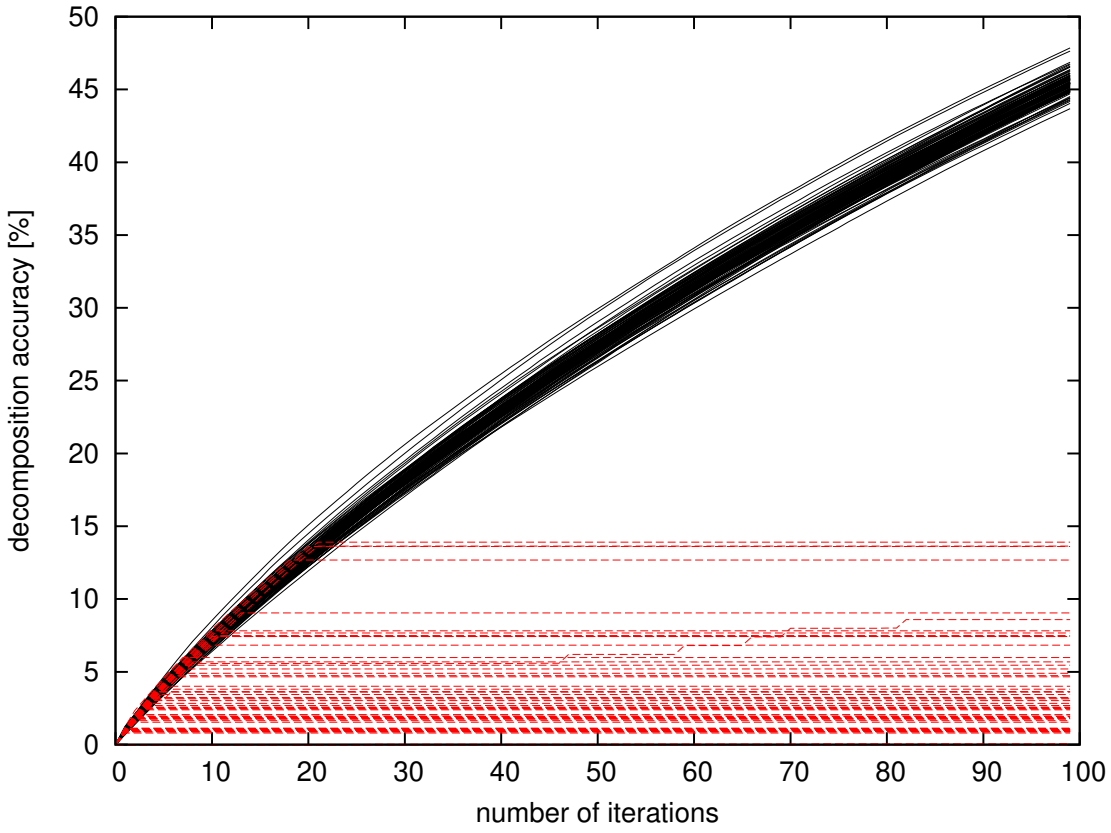


Figure 3: Decomposition quality for white noise signals for $K_r$-only (dashed red line) and correct (solid black line) normalization strategies.

## 3. Implementation

The normalization-aware version of MP algorithm has been implemented as an autonomous software package *empi*, written in C++ (2011 standard). The only external dependency is the FFTW library, chosen as the implementation of the Fast Fourier Transform due to its superior performance and compatibility with various architectures.

To take advantage of multi-threading (or multi-processor) architecture available currently in virtually all customer-grade personal computers, OpenMP parallelization has been introduced.

The software (current version: 0.4.1) is available from `https://github.com/develancer/empi` under a General Public Licence (version 3) as both C++

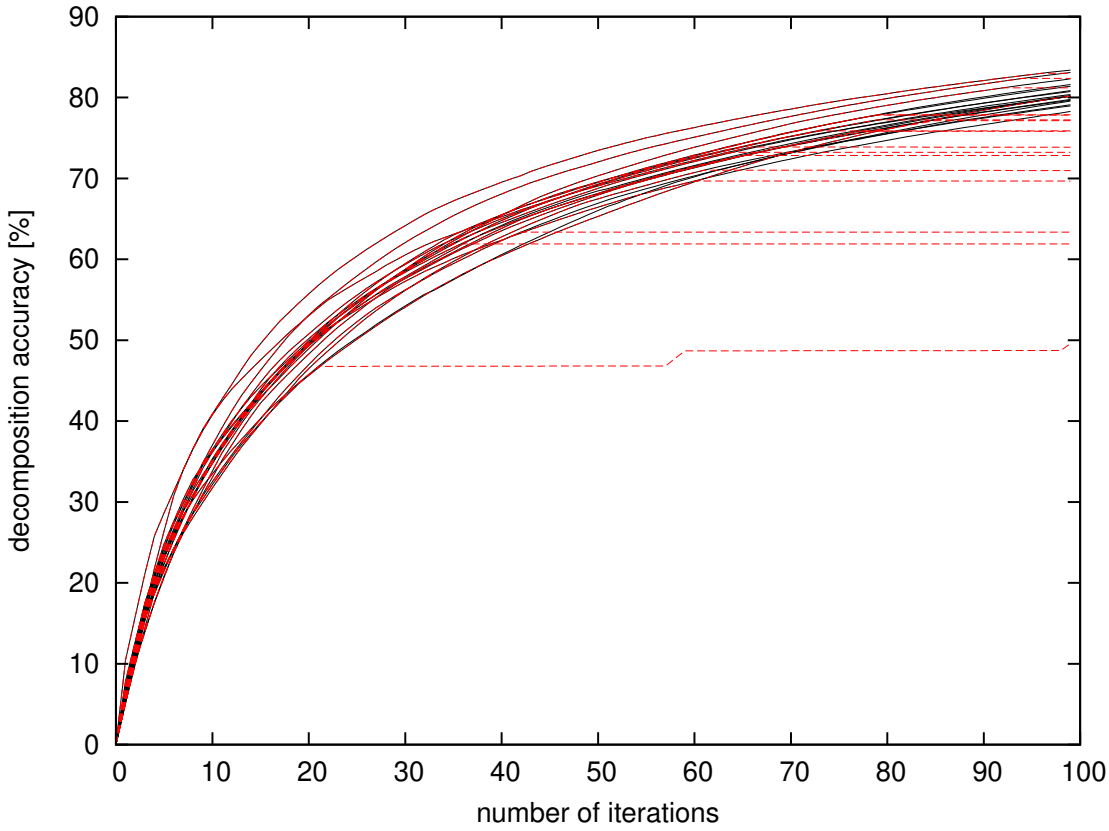


Figure 4: Decomposition quality for EEG signal segments for $K_r$-only (dashed red line) and correct (solid black line) normalization strategies.

source code and precompiled binaries for Linux and Microsoft Windows[1].

Apart from the correct treatment of atom normalization, a number of optimizations were implemented in *empi*. Some of the optimizations can be applied not only to matching pursuit with Gabor dictionaries, but to much wider spectrum of signal analysis algorithms (e.g. wavelet analysis), and therefore will be described in the following section.

## 3.1. Fast Fourier Transform

We introduce *complex* Gabor atoms as

$$G(s, f_0, t_0) = K_c \exp\left(-\pi\left(\frac{t - t_0}{s}\right)^2 + 2\pi \mathrm{i} f_0 (t - t_0)\right), \tag{13}$$

[1]both 32-bit and 64-bit OS versions are supported

where $K_c$ is a (much simpler) normalization constant, identical to the normalization factor of the envelope function alone:

$$K_c = \frac{2^{1/4}}{\sqrt{s}}\,. \tag{14}$$

The relation between complex and real Gabor atoms is pretty straightforward:

$$g(s, f_0, t_0, \phi) = \frac{K_r}{K_c}\,\Re(G(s, f_0, t_0)e^{\mathrm{i}\phi})\,, \tag{15}$$

where $\Re$ stands for the real part of the complex number. This simple substitution allows to perform a major part of the computations on complex Gabor atoms.

This approach allows us to make an important optimization. The full formula for the scalar product of the signal with a complex Gabor atom can be written as

$$\langle x, G(s, f_0, t_0)\rangle = K_c \sum_t x(t) e^{-\pi\left(\frac{t-t_0}{s}\right)^2} e^{2\pi\mathrm{i} f_0 (t-t_0)}\,, \tag{16}$$

which is essentially the formula for a windowed discrete Fourier transform with a Gaussian window function. Therefore, the calculations may be performed with Fast Fourier Transform implementation, resulting in a significant decrease of the computation time. It is worth noting, that this technique may be applied not only to Gabor atoms, but any family of atoms consisting of an oscillating factor and the "envelope" factor with finite support.

## 3.2. Phase optimization

For given $(s, f_0, t_0)$, the optimal phase $\phi_{\mathrm{opt}}$ maximizing the inner product can be calculated as

$$\phi_{\mathrm{opt}} = \arg\langle x, G(s, f_0, t_0)\rangle\,. \tag{17}$$

With such phase,

$$\left|\left\langle x, g(s, f_0, t_0, \phi_{\mathrm{opt}})\right\rangle\right| = \frac{K_r}{K_c}\,|\langle x, G(s, f_0, t_0)\rangle|\,. \tag{18}$$

Therefore, it is not necessary to include phase parameter in dictionary construction, as the optimal phase $\phi_{\mathrm{opt}}$ can be found in a straightforward manner for any given set of parameters $(s, f_0, t_0)$.

### 3.3. Priority queue

To improve the selection of the optimal atom in each step, similarly to the approach used in [7], a priority queue was introduced to store (and update) current values of $\langle R_n x, g\rangle$ for each atom $g$ in the dictionary. In each iteration, the best fit can be selected in $O(1)$ time by the Find-Max operation introduced in the standard Max-Heap implementation from [8].

After selecting an optimal atom in each iteration, values stored in the priority queue have to be updated, with the help of an additional Decrease-Key operation. However, from the formula

$$\langle R_{n+1}x, g\rangle = \langle R_n x - \alpha g_n, g\rangle = \langle R_n x, g\rangle - \alpha\langle g_n, g\rangle \tag{19}$$

it is clear that the inner products between updated residual and all atoms $g$ orthogonal to currently selected $g_n$ ($\langle g_n, g\rangle \approx 0$) do not require to be recalculated between iterations.

This optimization is utilized by noticing that the absolute value of the inner product between Gabor atoms (both real and complex) is bounded by the inner product of their Gaussian envelopes. For every two complex Gabor atoms $G_1 = G(s_1, f_1, t_1)$ and $G_2 = G(s_2, f_2, t_2)$, this upper bound can be calculated as:

$$|\langle G_1, G_2\rangle| \leq \sqrt{\frac{2s_1 s_2}{s_1^2 + s_2^2}} \exp\left(\frac{-\pi(t_1 - t_2)^2}{s_1^2 + s_2^2}\right). \tag{20}$$

Therefore, if the above scalar product is estimated to be below given threshold (e.g. $10^{-15}$), the pair of atoms may be treated as orthogonal and no inner product update have to be performed.

### 3.4. Multivariate Matching Pursuit

In case of analysing multichannel data, which is usually the case in EEG analysis, it is useful to take into account possible relations between simultaneous data in different channels. Therefore, the resulting decomposition of each channel will depend also on the data in every other channel. More specifically: in every iteration, atoms selected for all channels share the same values of parameters $s$, $f_0$ and $t_0$. Three variants of multivariate MP were implemented, as defined in [6]:

- MMP1: in every iteration, atoms selected for all channels share the same phase $\phi$ and the parameters maximize the sum of the moduli of inner products $\sum_c |\langle R^n x_c, g(s, f_0, t_0, \phi)\rangle|$.

- MMP2: in every iteration, atoms selected for all channels share the same phase $\phi$ and the parameters maximize the sum of the inner products $\sum_c \langle R^n x_c, g(s, f_0, t_0, \phi)\rangle$.

- MMP3: in every iteration, atoms selected for all channels are allowed to have different phases, and the parameters maximize the sum of the moduli of inner products $\sum_c |\langle R^n x_c, g(s, f_0, t_0, \phi_c)\rangle|$ .

Due to the linearity of the inner product, decomposition in MMP2 variant can be performed on a single signal, constructed as a sum of all channels, since

$$\sum_c \langle R^n x_c, g(s, f_0, t_0, \phi)\rangle = \left\langle \left( \sum_c R^n x_c \right), g(s, f_0, t_0, \phi) \right\rangle .$$

This feature of MMP2 allows for a significant speed-up, compared to MMP1. After each iteration, the selected atom $g_n$ has to be projected onto every channel to calculate the coefficients $\alpha$ (see eq. 1) for each channel. This additional step, however, is not computationally expensive.

# 4. Conclusions

This paper studies the effect of atom normalization on the performance and reliability of the matching pursuit algorithm. It was shown that the incorrect treatment of normalization may impede both the numerical stability of the algorithm, as well as its key feature—selecting the optimal atom at each step. A semi-analytical normalization strategy has been evaluated and shown to be accurate with the relative error not exceeding $10^{-12}$. Also, a ready-to-use C++ implementation, combining the introduced normalization strategy with a range of described optimization, has been provided on an open-source licence.

# 5. Acknowledgements

This research was supported by the Polish National Science Centre (UMO-2015/17/B/ST7/03784).

# References

[1] Mallat, S. G. and Zhang, Z., *Matching pursuits with time-frequency dictionaries*, IEEE Transactions on Signal Processing, Vol. 41, No. 12, Dec 1993, pp. 3397–3415.

[2] Durka, P. J., Ircha, D., Neuper, C., and Pfurtscheller, G., *Time-frequency microstructure of event-related electro-encephalogram desynchronisation and synchronisation*, Medical and Biological Engineering and Computing, Vol. 39, 2001, pp. 315–321.

[3] Sieluzycki, C., Konig, R., Matysiak, A., Kus, R., Ircha, D., and Durka, P. J., *Single-Trial Evoked Brain Responses Modeled by Multivariate Matching Pursuit*, IEEE Transactions on Biomedical Engineering, Vol. 56, No. 1, Jan 2009, pp. 74–82.

[4] Durka, P. J., Matysiak, A., Montes, E. M., Sosa, P. V., and Blinowska, K. J., *Multichannel matching pursuit and EEG inverse solutions.* Journal of neuroscience methods, Vol. 148 1, 2005, pp. 49–59.

[5] Durka, P. J., Malinowska, U., Zieleniewska, M., O'Reilly, C., Różański, P. T., and Żygierewicz, J., *Spindles in Svarog: framework and software for parametrization of EEG transients*, Front. Hum. Neurosci., 2015.

[6] Kuś, R., Różański, P. T., and Durka, P. J., *Multivariate matching pursuit in optimal Gabor dictionaries: theory and software with interface for EEG/MEG via Svarog*, BioMedical Engineering OnLine, Vol. 12, No. 1, Sep 2013, pp. 94.

[7] Krstulovic, S. and Gribonval, R., *MPTK: Matching Pursuit made Tractable*, In: Proc. Int. Conf. Acoust. Speech Signal Process. (ICASSP'06), Vol. 3, Toulouse, France, May 2006, pp. III–496 – III–499.

[8] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C., *Introduction to Algorithms, Third Edition*, The MIT Press, 3rd ed., 2009.

# Artykuł C: *Effects of envelope and dictionary structure on the performance of matching pursuit*

Piotr T. Różański (2020). „Effects of envelope and dictionary structure on the performance of matching pursuit". W: *IET Signal Processing* 14.2, s. 89–96. DOI: **10.1049/iet-spr.2019.0246**

IET Signal Processing

*Research Article*

IET Journals
The Institution of Engineering and Technology

# Effects of envelope and dictionary structure on the performance of matching pursuit

ISSN 1751-9675
Received on 21st May 2019
Revised 19th September 2019
Accepted on 6th November 2019
E-First on 29th November 2019
doi: 10.1049/iet-spr.2019.0246
www.ietdl.org

*Piotr T. Różański*[1] ✉

[1]*College of Inter-Faculty Individual Studies in Mathematics and Natural Sciences (MISMaP), University of Warsaw, ul. Stefana Banacha 2C, 02-097 Warszawa, Poland*

✉ *E-mail: ptr@mimuw.edu.pl*

**Abstract:** Matching pursuit is a greedy algorithm for computing a decomposition of the given signal as a linear combination of elements from an over-complete set (a 'dictionary'). This study generalises the construction of an optimal dictionary to non-Gaussian envelope functions, and demonstrates that optimising the dictionary with regard to the envelope function has a significant effect on the decomposition accuracy. The dictionary construction is then evaluated and compared for a range of possible envelope functions. Based on the results, a novel 'meta-exponential' envelope function is proposed and it is shown that for any given decomposition accuracy, it allows one to reduce the dictionary size (hence the computational time) by ∼10% compared to the most common case of decomposition with Gabor dictionaries. More importantly, the results of a decomposition study on high-quality audio files are presented, confirming that both the choice of the envelope and adjusting the dictionary to a given envelope have a significant effect on the overall performance of matching pursuit.

## 1 Introduction

The main idea of a large class of signal decomposition problems is to represent a given signal $x(t)$ as a linear combination of predefined functions $g_i(t) \in D$ as

$$x(t) \simeq \sum_i \alpha_i g_i(t) \tag{1}$$

where the set $D$ is called a *dictionary*. Unless all the elements in $D$ are mutually orthogonal, the decomposition cannot be computed by exact methods, as the optimal linear combination may not be unique.

Matching pursuit is a method introduced by Mallat and Zhang [1] that offers a solution to this problem in terms of a greedy, iterative algorithm. In each iteration, the best-matching atom is selected and subtracted from the residual signal from the previous iteration, i.e.

$$\begin{aligned} x^{(0)} &= x \\ g_i &= \arg\max_{g \in D}(g \cdot x^{(i-1)}) \\ \alpha_i &= g_i \cdot x^{(i-1)} \\ x^{(i)} &= x^{(i-1)} - \alpha_i g_i, \end{aligned} \tag{2}$$

where $\cdot$ denotes the scalar product

$$g \cdot x = \int_{-\infty}^{\infty} g(t)x(t)\mathrm{d}t\,. \tag{3}$$

the algorithm has been since successfully applied to the analysis of brain signals (most recently [2–4]), the removal of artefacts in a range of biomedical signals (ballistocardiographic [5], vibroarthrographic [6], electroencephalographic ([7], among others), electrocardiographic [8]), parametric audio coding [9], image deconvolution [10], analysis of seismic data [11], and, also recently, colour digital hologram compression [12].

In the case of the decomposition of real-valued signals, functions included in the dictionary are usually constructed by scaling, translating and modulating a pre-defined envelope function $\Phi$, as follows:

$$g_{(s,f_0,t_0)}(t) = K\Phi\left(\frac{t-t_0}{s}\right)\cos(2\pi f_0(t-t_0)+\varphi) \tag{4}$$

where $K$ is the normalisation factor, resulting from $L^2$-normalisation:

$$\int_{-\infty}^{+\infty} g^2_{(s,f_0,t_0)}(t)\mathrm{d}t = 1. \tag{5}$$

As for the precise construction of the dictionary [the sampling of the three-dimensional parameter space $(s, f_0, t_0)$], the original paper [1] introduced a *dyadic* dictionary based on the exponential sampling of a scale parameter ($s = a^j$) and the scale-dependent sampling for time ($\Delta_t = \delta t \cdot s$) and frequency $\left(\Delta_f = \delta f / s\right)$, where $a$, $\delta t$ and $\delta f$ could be chosen arbitrarily. The phase parameter $\varphi$ need not be included in the parameter space, as for any given $(s, f_0, t_0)$ the optimal phase $\varphi_{\mathrm{opt}}$ maximising the scalar product $g \cdot x$ can always be easily recovered.

Based on the concept of the metric related to the scalar product, the dictionary spacing for Gabor dictionaries (i.e. in the case where $\Phi$ is a Gaussian envelope) has been refined in [13], so that the sampling in all three dimensions is determined by a single parameter $\varepsilon$, representing distance (in terms of the product-related metric) between adjacent elements in the dictionary.

However, in general, the matching pursuit decomposition of a signal can be performed not only with the Gaussian envelope function, but for any well-behaved real-valued function $\Phi(t)$. This is especially important if the aim of signal decomposition is to extract some inherent (e.g. physiological) features of the signal, and the choice of the appropriate envelope function may be inferred from the assumed properties of the analysed system, e.g. [14]. The example applications of non-Gaussian function to matching pursuit include damped sinusoidal functions [15] used for parametric audio coding [16] and compression of digital fault records [17], as well as the Ricker envelope used for decomposition of seismic signals [18].

It is therefore interesting whether the choice of the envelope function has a significant effect on sampling in parameter space,

and on the overall dictionary size – especially that the computation time of matching pursuit is linearly dependent on the total number of elements in the dictionary.

To investigate this effect, the optimal dictionary construction will first be introduced for an arbitrary $\Phi(t)$, and then evaluated and compared for a range of potential envelope functions (Sections 2 and 3, with resulting formulae summarised in the Appendix). Based on these analytical results, detailed numerical tests will be performed (Section 4) to demonstrate the effect of dictionary construction on the algorithm's performance. Finally, a 'meta-exponential' envelope function will be introduced and it will be shown (by comparing the decomposition results of high-quality audio recordings in Section 5) that both the choice of the envelope function as well as adequate adaptation of the dictionary construction have a significant impact on the quality of the resulting decomposition.

## 2 Dictionary construction

### 2.1 Definitions

It will be assumed that the envelope function $\Phi$ is a $L^2$-normalised real-valued function ($\mathbb{R} \to \mathbb{R}$), fulfilling

$$\int_{-\infty}^{+\infty} \Phi^2(t)\mathrm{d}t = 1. \tag{6}$$

In order to keep the derivations compatible with the dictionary construction for Gabor functions as defined in [13], it will also be required that

$$\int_{-\infty}^{+\infty} t^2\Phi^2(t)\mathrm{d}t = \frac{1}{4\pi}. \tag{7}$$

Similarly to the usual approach, we will first focus on the distribution of scale parameter ($s$), and then, for each scale, calculate the optimal spacing in time and frequency. We will follow the construction of a dyadic dictionary [1] in which consecutive scales form a geometric series:

$$s \in \{s_0, s_0a, s_0a^2, \dots\} \tag{8}$$

### 2.2 Scale sampling

Following the approach of [13], we will require that the atoms with consecutive scales will be close to each other in terms of the product-induced metric:

$$d(g_{(s,\,f_0,\,t_0)}, g_{(as,\,f_0,\,t_0)}) \le \varepsilon, \tag{9}$$

where

$$d(g_1, g_2) = \sqrt{1 - \int_{-\infty}^{+\infty} g_1(t)g_2(t)\mathrm{d}t}. \tag{10}$$

The parameter $\varepsilon$ quantifies the difference between consecutive elements in the dictionary, and will be referred to as the 'mismatch parameter', whereas the value of $\varepsilon^2$ has been so far referred to as 'energy error' [13].

Substituting (1) and (2) into (7) and using a high-frequency approximation [13] gives

$$\int_{-\infty}^{+\infty} \Phi(ta^{1/2})\Phi(ta^{-1/2})\mathrm{d}t \ge 1 - \varepsilon^2. \tag{11}$$

However, in order to simplify the construction of the dictionary, it is useful to perform a substitution

$$\lambda = \ln s \tag{12}$$

equivalent to

$$\Delta_\lambda = \ln a, \tag{13}$$

as it will allow to define linear step values ($\Delta_\lambda$, $\Delta_f$, $\Delta_t$) for all three parameters, thus resulting in

$$\int_{-\infty}^{+\infty} \Phi(te^{\Delta_\lambda/2})\Phi(te^{-\Delta_\lambda/2})\mathrm{d}t \ge 1 - \varepsilon^2. \tag{14}$$

Let us term the left-hand side of the above inequality as $A(\Delta_\lambda)$. Finding the optimal (maximal for given $\varepsilon$) spacing for scale parameter is therefore equivalent to solving the equation:

$$A(\Delta_\lambda) = \int_{-\infty}^{+\infty} \Phi(te^{\Delta_\lambda/2})\Phi(te^{-\Delta_\lambda/2})\mathrm{d}t = 1 - \varepsilon^2 \tag{15}$$

i.e. calculating

$$\Delta_\lambda = A^{-1}(1 - \varepsilon^2). \tag{16}$$

### 2.3 Time and frequency sampling

For a given scale, we can repeat the procedure to find the optimal time and frequency spacing of the dictionary. Now, we require that

$$d(g_{(s,\,f_0,\,t_0)}, g_{(s,\,f_0+\Delta_f,\,t_0)}) \le \varepsilon \tag{17}$$

and

$$d(g_{(s,\,f_0,\,t_0)}, g_{(s,\,f_0,\,t_0+\Delta_t)}) \le \varepsilon. \tag{18}$$

Analogically, the calculations lead to introducing two more equations for the optimal time and frequency spacing:

$$\int_{-\infty}^{+\infty} \Phi^2(t)\cos(2\pi\Delta_f st)\mathrm{d}t = 1 - \varepsilon^2 \tag{19}$$

and

$$\int_{-\infty}^{+\infty} \Phi\left(t + \frac{\Delta_t}{2s}\right)\Phi\left(t - \frac{\Delta_t}{2s}\right)\mathrm{d}t = 1 - \varepsilon^2. \tag{20}$$

By introducing two integrals

$$B(x) = \int_{-\infty}^{+\infty} \Phi^2(t)\cos(2\pi xt)\mathrm{d}t \tag{21}$$

and

$$C(x) = \int_{-\infty}^{+\infty} \Phi\left(t + \frac{1}{2}x\right)\Phi\left(t - \frac{1}{2}x\right)\mathrm{d}t, \tag{22}$$

finding the optimal time and frequency spacing can be related to solving two equations dependent on the envelope function:

$$B(\Delta_f s) = 1 - \varepsilon^2 \tag{23}$$

and

$$C\left(\frac{\Delta_t}{s}\right) = 1 - \varepsilon^2. \tag{24}$$

### 2.4 Normalisation

Substituting (4) into (5) allows one to derive a formula for the normalisation factor $K$. Taking into account the normalisation of the envelope itself (6), the equation reduces to

$$K = \sqrt{\frac{2}{s}}\left(1 + \cos(2\varphi)\int_{-\infty}^{+\infty}\Phi^2(t)\cos(4\pi f_0 st)\mathrm{d}t - \sin(2\varphi)\int_{-\infty}^{+\infty}\Phi^2(t)\sin(4\pi f_0 st)\mathrm{d}t\right)^{-1/2}. \quad (25)$$

Furthermore, for symmetric envelopes [i.e. $\Phi(-t) = \Phi(t)$], the second part in the above equation vanishes, and the normalisation factor can be written with the use of the function defined in (21) as

$$K = \sqrt{\frac{2}{s}}(1 + \cos(2\varphi)B(2f_0 s))^{-1/2}. \quad (26)$$

## 3 Examples for various envelopes

Based on the formulae from the previous section, it is possible to define the optimal dictionary construction for various envelope functions. This will allow one to compare different envelopes and examine the effect on the dictionary structure for all three parameters (scale, frequency and position). The normalised formulae, with the analytic forms of integrals $A$, $B$, and $C$ for a set of envelope functions, are included in the Appendix. The examined envelope functions include:

- rectangular envelope,
- triangular envelope,
- Gaussian envelope,
- exponential envelope,
- Lorentzian (Cauchy) envelope,
- cosine envelope,
- squared cosine envelope.

The rectangular envelope is included here mainly for a matter of consequence, as it has limited practical value. Performing the decomposition with a rectangular envelope would be equivalent to decomposing the signal into a sum of sine function fragments. Since the envelope function has a discontinuity at $t = \pm\sqrt{3/4\pi}$, subtracting such a function from the signal might introduce undesirable discontinuities in the residual, impeding further iterations of the decomposition process.

Also, it can be noted that albeit the variance normalisation introduced by (7) is aimed mainly at compatibility with the established definition of Gabor atoms, it nevertheless leads to the rectangular envelope having an approximate width of 1 ($\simeq 0.98$), consistent with the intuitive understanding of the 'scale' parameter. A similar observation can be made for the Gaussian envelope in its normalised form, as the full-width at half-maximum of this function ($\simeq 0.94$) is also close to unity.

### 3.1 Comparison of dictionary structures

Relations between step values ($\Delta_\lambda$, $\Delta_f$ and $\Delta_t$) and mismatch parameter $\varepsilon$ for examined envelope functions are presented in Fig. 1. Larger step values are preferred, as they allow for the coarser sampling in the parameter space. Several important observations can be made based on the presented data.

Firstly, it can be noticed that for all examined envelopes, the ratio $\Delta_f s/\varepsilon$ converges to the same value for a sufficiently dense dictionary ($\varepsilon \to 0$). This stems from the fact that for small values of $\varepsilon$, it is possible to approximate

$$B(\Delta_f s) = \int_{-\infty}^{+\infty}\Phi^2(t)\cos(2\pi\Delta_f st)\mathrm{d}t \simeq \int_{-\infty}^{+\infty}\Phi^2(t)\left(1 - \frac{1}{2}(2\pi\Delta_f st)^2\right)\mathrm{d}t \quad (27)$$

which, by (6) and (7), can be reduced to

$$B(\Delta_f s) \simeq 1 - \frac{\pi}{2}\Delta_f^2 s^2, \quad (28)$$

which, combined with (23), gives

$$\lim_{\varepsilon \to 0}\frac{\Delta_f s}{\varepsilon} = \sqrt{\frac{2}{\pi}} \simeq 0.798. \quad (29)$$

As for the results of particular envelope functions, the rectangular envelope has the worst performance in all three dimensions. Apart from the already mentioned difficulties associated with decomposition with the rectangular envelope, the resulting dictionary would have to be much larger (up to 10–15 times for small values of $\varepsilon$) compared to the other envelope functions.

On the other hand, the Gaussian envelope (leading to the dictionary of Gabor atoms) has comparably good performance for all three parameters, resulting in the very good overall outcome, especially for dense dictionaries ($\varepsilon \to 0$). However, it is somewhat surprising (in the context of the widespread use of Gabor dictionaries) that it does not yield the best performance in a wider range of $\varepsilon$.

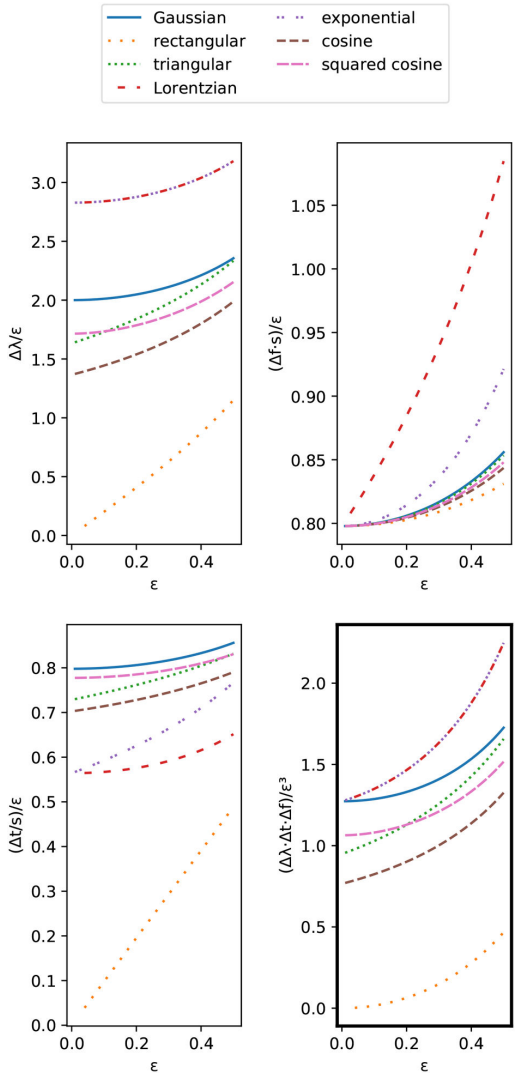


**Fig. 1** *Relation between the mismatch parameter ε (on all horizontal axes) and the normalised step values for all three dimensions. Larger values on vertical axes allow for coarser sampling for the given parameter. The last plot (on the bottom right) presents products of values in the preceding plots, inversely proportional to the size of the dictionary*

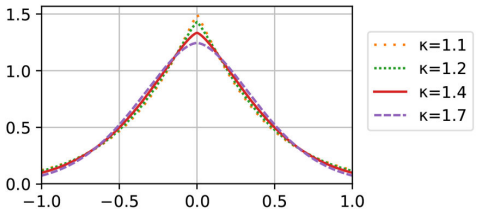


**Fig. 2** *Shapes of the optimised 'meta-exponential' envelope functions for various values of* $\kappa$

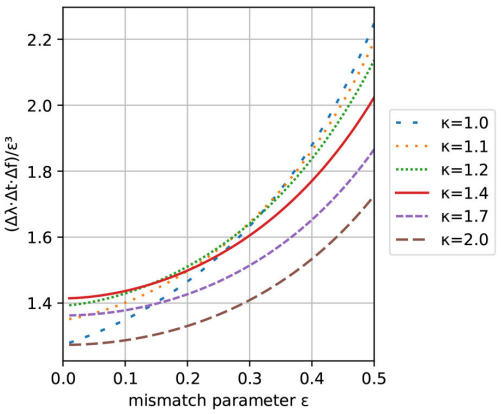


**Fig. 3** *Relation between the mismatch parameter* $\varepsilon$ *and the product of normalised step values for all three dimensions (inversely proportional to the size of the dictionary), calculated for a range of 'meta-exponential' envelope functions*

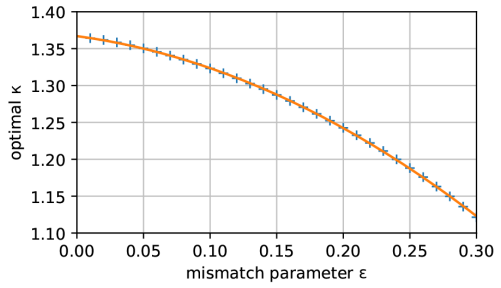


**Fig. 4** *Numerically calculated optimal (in terms of a dictionary size) parameter* $\kappa$ *for 'meta-exponential' envelopes, as a function of mismatch parameter* $\varepsilon$*, with a least-squares polynomial fit (see the text)*

The best overall performance is achieved by exponential and Lorentzian envelope functions. Both functions perform exactly the same for the scale parameter, and in terms of the overall performance, as these two functions are related to each other through the Fourier transform. This is also reflected by the exact analytical form of the $I$ integral in (51)≡(55), as well as the striking symmetry in the relations between $B$ and $C$ integrals for these envelopes: (52) and (53) versus (56) and (57).

However, for practical reasons, the Lorentzian envelope is not well suited for signal decomposition because of its tail properties, as it would require to evaluate the function and compute the scalar product with the signal on the interval of width $\simeq 20\,s$, to capture the most of the signal energy (at the relative error of $10^{-6}$). For comparison, the exponential envelope, at the same threshold, requires the width of only $\simeq 4.6\,s$. Therefore, among the examined envelope functions, the exponential envelope seems to be most promising.

### 3.2 Search for the optimised envelope function

The results of the previous section clearly state that the best performance is provided by exponential and Gaussian envelopes. Based on this observation, it is justified to ask whether it is possible to construct an even better performing function as a generalisation of those slow-vanishing envelopes.

Let us consider a 'meta-exponential' family of functions

$$\Phi_\kappa(t) = \alpha_\kappa \exp(-\beta_\kappa |t|^\kappa)\,. \tag{30}$$

For $\kappa > 2$, such functions are also referred to as 'super-Gaussian' [19]. It should be noted, that as long as $\kappa > 1$, proposed envelope functions are both continuous and differentiable on the entire real axis, including $t = 0$. Introducing the normalisation defined in (6) and (7) gives

$$\alpha_\kappa = \sqrt[4]{\pi\kappa^2 \frac{\Gamma(3/\kappa)}{\Gamma(1/\kappa)^3}} \tag{31}$$

and

$$\beta_\kappa = 2^{\kappa-1}\left(\pi \frac{\Gamma(3/\kappa)}{\Gamma(1/\kappa)}\right)^{\kappa/2} \tag{32}$$

where $\Gamma$ stands for the Euler integral of the second kind:

$$\Gamma(z) = \int_0^\infty x^{z-1}\mathrm{e}^{-x}\mathrm{d}x\,. \tag{33}$$

Plots of the proposed family of envelope functions, for values of $\kappa$ ranging between 1 (corresponding to the purely exponential function) and 2 (representing the Gaussian function), are presented in Fig. 2. For such functions, integral $A$ can be written as

$$A(\Delta_\lambda) = \cosh\left(\frac{\kappa}{2}\Delta_\lambda\right)^{-1/k}, \tag{34}$$

whereas the other two integrals ($B$ and $C$) have to be calculated numerically. This, however, does not pose any threat to the performance of matching pursuit implementations, as CPU time spent on generating the dictionary is usually negligible compared to the main (iterative) stage of the computation.

The performance resulting from using such envelopes for dictionary construction is presented in Fig. 3, as a function of $\varepsilon$. The results show that the optimal value of $\kappa$ depends on $\varepsilon$, while the pure exponential ($\kappa = 1$) envelope is optimal for very coarse dictionaries, the increase of the needed precision also increases the optimal exponent, up to the value of $\kappa \simeq 1.4$. Values higher than that (e.g. $\kappa = 1.7$ and $\kappa = 2.0$ on the plot) do not seem to be optimal for any value of $\varepsilon$.

For a more constructive result, it is possible to numerically calculate the optimal exponent $\kappa$ as a function of $\varepsilon$. Results of such a computation are presented in Fig. 4, together with a least-squares quadratic fit approximating the optimal value of $\kappa$ as

$$\kappa_{\mathrm{opt}} \simeq 1.367 - 0.244\varepsilon - 1.898\varepsilon^2\,. \tag{35}$$

## 4 Numerical evaluation

### 4.1 Evaluating the significance of the dictionary construction

The first issue in need for a numerical test is whether the described optimisation of the dictionary construction for some arbitrary envelope function is significant in terms of the performance of matching pursuit. To deal with this issue, two rounds of decomposition were performed, both with triangular envelope function. However, while the first round used a dictionary optimised for the triangular envelope as in Section 8.2.2, the second round used a dictionary as it would be for Gabor atoms [13].

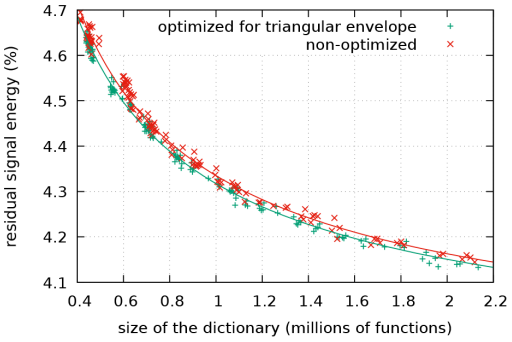


**Fig. 5** *Average residual energy of the decomposition with triangular envelopes, as a function of dictionary size, comparing two types of dictionary construction (see the text)*

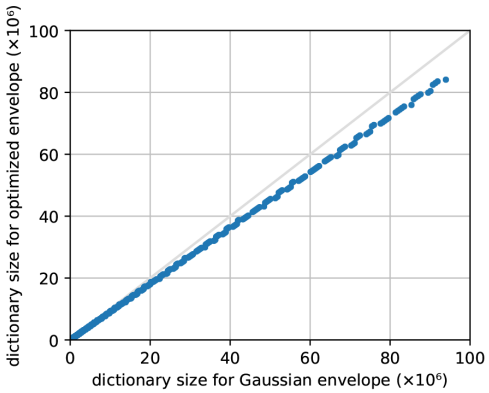


**Fig. 6** *Size of the generated dictionary for Gaussian and 'meta-exponential' envelope functions, for different values of mismatch parameter $\varepsilon$*

The decomposition in each round was performed for one thousand 5-s epochs of synthetic signal (at the sampling frequency of 512 Hz). Each epoch consisted of a single normalised oscillating function $g(t)$, as defined in (4), of randomly chosen parameters $s$, $f_0$, $t_0$ (independent of the actual realisation of the dictionary) with a Gaussian noise added at a signal-to-noise ratio of 25. For each round, the average residual energy was computed after performing one iteration of matching pursuit per epoch.

Each of the two rounds of decomposition, as described above, was performed for a range of possible values of $\varepsilon$ between 0.07 and 0.14. The relation between the size of the dictionary (linearly corresponding to the computation time) and the average residual energy is presented in Fig. 5, together with the rudimentary exponential fit approaching the value of $1/(\mathrm{SNR}+1) \simeq 3.85\%$, corresponding to the theoretical perfect decomposition of the signal.

### *4.2 Evaluating the dictionary size for the optimised envelope*

Since the computation time of matching pursuit is linearly dependent on the size of the dictionary, it is reasonable to ask whether using an optimised 'meta-exponential' envelope (as introduced in Section 3.2) instead of the Gaussian envelope results in a significantly smaller dictionary size in a realistic use case.

To verify this issue, a dictionary resulting from both envelope functions was calculated for a range of possible values of $\varepsilon$ between 0.03 and 0.15. The sampling frequency of 512 Hz was assumed, and the frequency range covered by the dictionary was from 0 to 40 Hz (similarly to the case of performing the decomposition of low-pass filtered signals). The scale parameter ($s$) was constrained to the range of 0.1–5 s, and the epoch length was set at 20 s. The treatment of boundary conditions is described in Section 9.1 of the Appendix.

Calculated dictionary sizes for both envelopes, as a function of $\varepsilon$, are presented in Fig. 6, together with an identity function for comparison. The results show that using the optimised 'meta-exponential' envelope allows for a ∼10% reduction of the dictionary size (and therefore, computational time) at the same level of accuracy defined by the mismatch parameter $\varepsilon$.

## 5 Application to parametric audio coding

In addition to the numerical results from the previous section, it is reasonable to ask whether the choice of the envelope and dictionary construction has any significant effect on the overall performance of matching pursuit in real-world applications. To answer this question, a series of matching pursuit decompositions was computed for a set of publicly available high-quality audio files [20]. Three variants were evaluated:

1. a decomposition with Gabor atoms and standard optimal dictionary as defined in [13],
2. a decomposition with a meta-exponential envelope with $\kappa = 4$ and the same dictionary as in point 1,
3. a decomposition with a meta-exponential envelope with $\kappa = 4$ and a dictionary adapted to this specific envelope.

Parameter $\kappa = 4$ for the envelope function was chosen due to two favourable qualities of such an envelope for audio decomposition: 'flatter' central part of the envelope and faster decay outside of this central part. However, this choice is somewhat arbitrary, as the main purpose of this experiment is not to find the 'best' envelope function for audio applications, but rather to assess how the construction of the dictionary might affect the performance of the algorithm.

The parameters $\varepsilon$ were adjusted to equalise the sizes of the dictionaries for all three variants. However, due to the discrete nature of the dictionary construction it was not possible to obtain the equal number of atoms – instead, the dictionary size for the variant no. 3 was deliberately set to be smaller than the other two. Target decomposition accuracy was set to 95% (meaning that the decomposition aimed to reduce the residual energy below 5% of the total signal energy), and the number of iterations needed to achieve this threshold was taken as the performance measure of the decomposition. This value was chosen as the performance measure since the number of atoms in the resulting decomposition directly corresponds to the achieved compression ratio.

Three audio files, representing different styles of music, were selected for the decomposition from the top of the list [20] (the labels are taken verbatim from the source webpage):

- 2L-106) Arnesen: MAGNIFICAT 4. Et misericordia,
- 2L-139) Chromatic Fantasia and Fugue in D minor, BWV 903: Fantasia,
- 2L-145) Hoff: Innocence.

The middle 60-s part was extracted from each of these audio files (in 'Original CD' quality of 16-bit 44,100 Hz sampling) and divided into 1-s epochs. The dictionary was limited to atoms with a scale between 5 and 500 ms, with frequency not exceeding 20,000 Hz. Only the first (left) channel of each stereo recording was used.

The results of this experiment are presented in Table 1. Choosing a meta-exponential envelope instead of the Gaussian envelope has been shown to reduce the decomposition size by ∼5%. Moreover, although the dictionary in variant 3 was smaller than in the other two, it provided an additional size reduction for all three audio files, thus confirming the importance of adjusting the dictionary structure to the envelope function.

## 6 Conclusions

The generalised construction of the dictionary has been discussed and calculated for a range of possible envelope functions, allowing to construct an optimal dictionary for any given envelope function

**Table 1** Audio coding performance for various audio files and dictionary variants (see the text)

| Dictionary size | Variant 1 | Variant 2 | Variant 3 |
|---|---|---|---|
| | 43414929 | | 43349135 |
| Audio file | Total number of iterations (±relative to variant 1) | | |
| 2L-106 | 13,691 | 13,134 (−4.1%) | 13,110 (−4.2%) |
| 2L-139 | 3090 | 2923 (−5.4%) | 2915 (−5.7%) |
| 2L-145 | 5742 | 5437 (−5.3%) | 5431 (−5.4%) |

(e.g. adapted to a specific problem). It has been shown that adjusting the construction to a specific envelope function has a significant effect on the quality of the resulting decomposition.

Moreover, it has been demonstrated that it is possible to reduce the computational time of matching pursuit decomposition while preserving the same precision (defined by the mismatch parameter $\varepsilon$), by using the optimised 'meta-exponential' function instead of the Gaussian envelope for the construction of the dictionary. It has been shown as well that envelope functions of this kind can exceed the performance of Gabor dictionaries in the decomposition of audio signals.

The proposed construction can be naturally applied to all variants of matching-pursuit-based algorithms, e.g. orthogonal matching pursuit [21] and its variations, constrained matching pursuit [22], as well as various gradient pursuits [23].

## 7 Acknowledgments

The author thanks prof. Piotr Durka for many valuable suggestions. Numerical calculations in Section 4.1 were performed using open-source matching pursuit implementation *empi* (https://github.com/develancer/empi), and the source code adapted for these calculations (based on version 0.4.3) is available on a branch 'test-tri' (tagged as *test-tri-reference* and *test-tri-optimised*) in the source code repository. Similarly, the calculations in Section 5 were based on the versions of code tagged as *test-variant2* and *test-variant3*. The boundary conditions treatment described in Section 8.1 is consistent with the referenced version of the software.

## 8 References

[1] Mallat, S.G., Zhang, Z.: 'Matching pursuits with time-frequency dictionaries', *IEEE Trans. Signal Process.*, 1993, **41**, (12), pp. 3397–3415
[2] Seco, G.B.S., Gerhardt, G.J.L., Biazotti, A.A., *et al.*: 'EEG alpha rhythm detection on a portable device', *Biomed. Signal Proc. Control*, 2019, **52**, pp. 97–102
[3] Zieleniewska, M., Duszyk, A., Różański, P., *et al.*: 'Parametric description of EEG profiles for assessment of sleep architecture in disorders of consciousness', *Int. J. Neural Syst.*, 2018, **29**, (3), p. 1850049
[4] Larocco, J., Franaszczuk, P.J., Kerick, S., *et al.*: 'Spindler: a framework for parametric analysis and detection of spindles in EEG with application to sleep spindles', *J. Neural Eng.*, 2018, **15**, p. 066015
[5] Xia, H., Ruan, D., Cohen, M.S.: 'Removing ballistocardiogram (BCG) artifact from full-scalp EEG acquired inside the MR scanner with orthogonal matching pursuit (OMP)', *Front. Neurosci.*, 2014, **8**, pp. 1–12
[6] Wu, Y., Yang, S., Zheng, F., *et al.*: 'Removal of artifacts in knee joint vibroarthrographic signals using ensemble empirical mode decomposition and detrended fluctuation analysis', *Physiol. Meas.*, 2014, **35**, pp. 429–439
[7] Schneider, R., Lau, S., Kuhlmann, L., *et al.*: 'Matching pursuit based removal of cardiac pulse-related artifacts in EEG/fMRI', World Academy of Science, Engineering and Technology, Paris, France, November 2011
[8] Husøy, J.H., Eilevstjønn, J., Eftestøl, T., *et al.*: 'Removal of cardiopulmonary resuscitation artifacts from human ECG using an efficient matching pursuit-like algorithm', *IEEE Trans. Biomed. Eng.*, 2002, **49**, pp. 1287–1298
[9] Yang, J., He, Q., Li, Y., *et al.*: 'Dictionary learning based on m-pca-n for audio signal sparse representation', *IET Signal Process.*, 2018, **12**, (2), pp. 198–206
[10] Gong, D., Tan, M., Shi, Q., *et al.*: 'MPTV: matching pursuit-based total variation minimization for image deconvolution', *IEEE Trans. Image Process.*, 2019, **28**, pp. 1851–1865
[11] Li, C., Zhang, F.: 'Matching pursuit parallel decomposition of seismic data', *Comput. Geosci.*, 2017, **104**, pp. 54–61
[12] Rhammad, A.E., Gioia, P., Gilles, A., *et al.*: 'Color digital hologram compression based on matching pursuit', *Appl. Opt.*, 2018, **57**, (17), pp. 4930–4942
[13] Kuś, R., Tadeusz Różański, P., Durka, P.J.: 'Multivariate matching pursuit in optimal Gabor dictionaries: theory and software with interface for EEG/MEG via Svarog', *Biomed. Eng. Online*, 2013, **12**, (1), p. 94
[14] Sun, R.-B., Yang, Z.-B., Zhai, Z., *et al.*: 'Sparse representation based on parametric impulsive dictionary design for bearing fault diagnosis', *Mech. Syst. Signal Process.*, 2019, **122**, pp. 737–753
[15] Goodwin, M.: 'Matching pursuit with damped sinusoids'. 1997 IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, Munich, Germany, April 1997, vol. 3, pp. 2037–2040
[16] Derrien, O., Badeau, R., Richard, G.: 'Parametric audio coding with exponentially damped sinusoids', *IEEE Trans. Audio, Speech, Lang. Process.*, 2013, **21**, pp. 1489–1501
[17] Tcheou, M.P., Miranda, A.L.L., Lovisolo, L., *et al.*: 'How far can one compress digital fault records? Analysis of a matching pursuit-based algorithm', *Dig. Signal Process. Rev. J.*, 2012, **22**, pp. 288–297
[18] Zhang, F., Liu, H., Dai, R.: 'Exponential pursuit algorithm based on Ricker wavelet for seismic signal decomposition', *Zhongguo Kuangye Daxue Xuebao/J. China Univ. Mining Technol.*, 2016, **45**, pp. 128–132
[19] Parent, A., Morin, M., Lavigne, P.: 'Propagation of super-Gaussian field distributions', *Opt. Quantum Electron.*, 1992, **24**, (9), pp. S1071–S1079
[20] Lindberg, M.: '2L high resolution music (Lindberg Lyd AS)'. http://www.2l.no/hires/index.html, accessed 30 July 2019
[21] Pati, Y.C., Rezaiifar, R., Krishnaprasad, P.S.: 'Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition'. Proc. of 27th Asilomar Conf. on Signals, Systems and Computers, Pacific Grove, CA, USA, 1993, vol. 1, pp. 40–44
[22] Adler, A., Emiya, V., Jafari, M.G., *et al.*: 'A constrained matching pursuit approach to audio declipping'. 2011 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), Prague, Czech Republic, May 2011, pp. 329–332
[23] Blumensath, T., Davies, M.E.: 'Gradient pursuits', *IEEE Trans. Signal Process.*, 2008, **56**, (6), pp. 2370–2382

## 9 Appendix

### 9.1 Boundary conditions of dictionary parameters

Boundary conditions in the parameter space were selected so that both boundary values for each parameter would always be represented in the dictionary. For each of the three parameters ($x \in \{\lambda, t, f\}$), given the boundary values $x_{\min}$ and $x_{\max}$ and maximal step value $\Delta_x$ resulting from the dictionary construction, the minimal number of steps is computed as

$$N = \left\lceil \frac{x_{\max} - x_{\min}}{\Delta_x} \right\rceil, \tag{36}$$

and then the parameter values $x_0, x_1, \ldots, x_N$ for the dictionary are computed as

$$x_i = x_{\min} + \frac{i}{N}(x_{\max} - x_{\min}), \tag{37}$$

thus giving a total of $N + 1$ values in the given dimension.

### 9.2 Formulae for various envelope functions

All of the functions described in this section are normalised according to (6) and (7). Several formulae include the function sinc, defined as

$$\operatorname{sinc}(x) = \frac{\sin(\pi x)}{\pi x}. \tag{38}$$

Envelope functions are limited to $t \in [-1; 1]$ solely for the purpose of plotting, as this interval is in no way imposed on the calculations.

#### 9.2.1 Rectangular envelope:

$$\Phi(t) = \begin{cases} \left(\frac{\pi}{3}\right)^{1/4}, & -\sqrt{\frac{3}{4\pi}} \le t \le \sqrt{\frac{3}{4\pi}}, \\ 0 & \text{otherwise} \end{cases} \tag{39}$$

For the rectangular envelope (see Fig. 7), integrals $A$, $B$ and $C$ are defined as:

$$A(\Delta_\lambda) = \mathrm{e}^{-(1/2)\Delta_\lambda} \tag{40}$$

$$B(x) = \mathrm{sinc}\left(\sqrt{\frac{3}{\pi}}x\right) \tag{41}$$

$$C(x) = \begin{cases} 1 - \sqrt{\frac{\pi}{3}}|x|, & |x| \le \sqrt{\frac{3}{\pi}} \\ 0 & \text{otherwise} \end{cases} \tag{42}$$

#### 9.2.2 Triangular envelope:

$$\Phi(t) = \begin{cases} \left(\frac{9\pi}{10}\right)^{1/4}\left(1 - |t|\sqrt{\frac{2\pi}{5}}\right), & -\sqrt{\frac{5}{2\pi}} \le t \le \sqrt{\frac{5}{2\pi}}, \\ 0 & \text{otherwise} \end{cases} \tag{43}$$

For the triangular envelope (see Fig. 8), integrals $A$, $B$ and $C$ are defined as:

$$A(\Delta_\lambda) = \frac{3}{2}\mathrm{e}^{-(1/2)\Delta_\lambda} - \frac{1}{2}\mathrm{e}^{-(3/2)\Delta_\lambda} \tag{44}$$

$$B(x) = \frac{3 \cdot (1 - \mathrm{sinc}(\sqrt{(10/\pi)}x))}{5\pi x^2} \tag{45}$$

$$C(x) = \begin{cases} 1 - \frac{3\pi}{5}x^2 + \sqrt{\frac{9\pi^3}{250}}|x|^3, & |x| \le \sqrt{\frac{5}{2\pi}} \\ 2 \cdot \left(1 - |x|\sqrt{\frac{\pi}{10}}\right)^3, & \sqrt{\frac{5}{2\pi}} < |x| \le \sqrt{\frac{10}{\pi}} \\ 0 & \text{otherwise} \end{cases} \tag{46}$$

#### 9.2.3 Gaussian envelope:

$$\Phi(t) = 2^{1/4}\mathrm{e}^{-\pi t^2} \tag{47}$$

For the Gaussian envelope (see Fig. 9), integrals $A$, $B$ and $C$ are defined as:

$$A(\Delta_\lambda) = (\cosh \Delta_\lambda)^{-1/2} \tag{48}$$

$$B(x) = C(x) = \mathrm{e}^{-(1/2)\pi x^2} \tag{49}$$

#### 9.2.4 Exponential envelope:

$$\Phi(t) = (2\pi)^{1/4}\mathrm{e}^{-\sqrt{2\pi}|t|} \tag{50}$$

For the exponential envelope (see Fig. 10), integrals $A$, $B$ and $C$ are defined as:

$$A(\Delta_\lambda) = \left(\cosh\frac{\Delta_\lambda}{2}\right)^{-1} \tag{51}$$

$$B(x) = \frac{1}{1 + (\pi/2)x^2} \tag{52}$$

$$C(x) = (1 + \sqrt{2\pi}|x|)\mathrm{e}^{-\sqrt{2\pi}|x|} \tag{53}$$

#### 9.2.5 Lorenzian envelope:

$$\Phi(t) = \frac{2\pi^{-1/4}}{1 + 4\pi t^2} \tag{54}$$

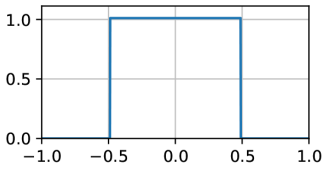


**Fig. 7** *Rectangular envelope*

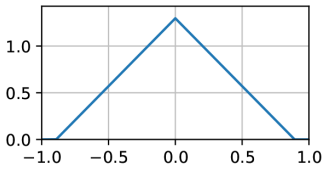


**Fig. 8** *Triangular envelope*

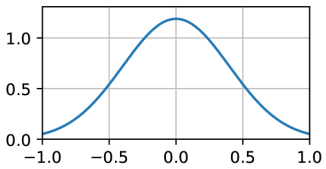


**Fig. 9** *Gaussian envelope*

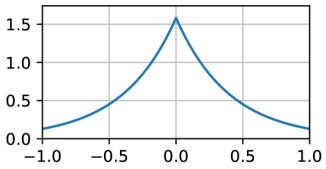


**Fig. 10** *Exponential envelope*

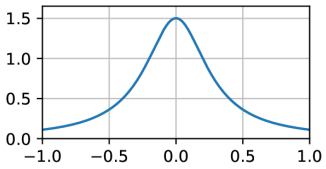


**Fig. 11** *Lorenzian envelope*

For the Lorentzian envelope (see Fig. 11), integrals $A$, $B$ and $C$ are defined as:

$$A(\Delta_\lambda) = \left(\cosh\frac{\Delta_\lambda}{2}\right)^{-1} \tag{55}$$

$$B(x) = (1 + \sqrt{\pi}|x|)\mathrm{e}^{-\sqrt{\pi}|x|} \tag{56}$$

$$C(x) = \frac{1}{1 + \pi x^2} \tag{57}$$

#### 9.2.6 Cosine envelope:

$$\Phi(t) = \begin{cases} \sqrt{\frac{2\Omega_1}{\pi}}\cos(\Omega_1 t), & -\frac{\pi}{2\Omega_1} \le t \le \frac{\pi}{2\Omega_1} \\ 0 & \text{otherwise} \end{cases} \tag{58}$$

where $\Omega_1 = \sqrt{(\pi^3/3) - 2\pi}$.

For the cosine envelope (see Fig. 12), integrals $A$, $B$ and $C$ are defined as:

$$A(\Delta_\lambda) = \frac{2}{\pi} e^{(1/2)\Delta_\lambda} \frac{\cos\left((\pi/2)e^{-\Delta_\lambda}\right)}{\sinh(\Delta_\lambda)} \tag{59}$$

$$B(x) = \frac{\operatorname{sinc}(\pi x/\Omega_1)}{1-(\pi x/\Omega_1)^2} \tag{60}$$

$$C(x) = \begin{cases} \frac{1}{\pi}\sin(\Omega_1 x) + \left(1 - \frac{\Omega_1 x}{\pi}\right)\cos(\Omega_1 x), & |x| \le \frac{\pi}{\Omega_1} \\ 0 & \text{otherwise} \end{cases} \tag{61}$$

### 9.2.7 Squared cosine envelope:

$$\Phi(t) = \begin{cases} \sqrt{\frac{8\Omega_2}{3\pi}}\cos^2(\Omega_2 t), & -\frac{\pi}{2\Omega_2} \le t \le \frac{\pi}{2\Omega_2} \\ 0 & \text{otherwise} \end{cases} \tag{62}$$

where $\Omega_2 = \sqrt{(\pi^3/3) - \frac{15}{6}\pi}$.

For the squared cosine envelope (see Fig. 13), integrals $A$, $B$ and $C$ are defined as:

$$A(\Delta_\lambda) = \frac{1}{3} e^{(1/2)\Delta_\lambda} \frac{\operatorname{sinc}(e^{-\Delta})}{\sinh(\Delta)} + \frac{2}{3} e^{-(1/2)\Delta_\lambda} \tag{63}$$

$$B(x) = \frac{\operatorname{sinc}(\pi x/\Omega_2)}{1-(5/4)(\pi x/\Omega_2)^2 + (1/4)(\pi x/\Omega_2)^4} \tag{64}$$

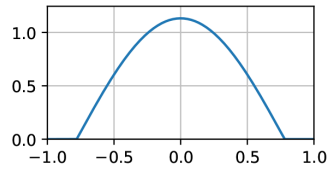


**Fig. 12** *Cosine envelope*

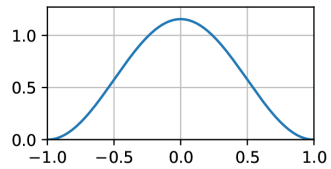


**Fig. 13** *Squared cosine envelope*

$$C(x) = \begin{cases} \frac{1}{2\pi}\sin(2\Omega_2 x) + \frac{1}{3}\cdot(1 - \frac{\Omega_2 x}{\pi})\cos(2\Omega_2 x) - \frac{2\Omega_2 x}{3\pi} + \frac{2}{3}, & |x| \le \\ 0 & \text{otherw} \end{cases} \tag{65}$$

# Artykuł D: *empi: GPU-accelerated matching pursuit with continuous dictionaries*

Piotr T. Różański (2024). „*empi*: GPU-accelerated matching pursuit with continuous dictionaries". W: *ACM Trans. Math. Softw.* (przyjęte do druku)

# empi: GPU-accelerated matching pursuit with continuous dictionaries

PIOTR T. RÓŻAŃSKI, University of Warsaw, Poland

This article introduces an effective approach to performing matching pursuit calculations with continuous (quasi-infinite) dictionaries. Simulating continuous parameter space is accomplished by combining optimal dictionary construction as introduced previously with local parameter optimization. At the same time, the calculations can be performed not only with Gabor atoms, but with any type of oscillating atoms, or even multiple types of atoms at once. The proposed method is confirmed by introducing a first stable version of a high-performance implementation *empi*, supporting calculations with continuous as well as discrete dictionaries, further accelerated by the use of multi-threading and GPU support.

CCS Concepts: • **Mathematics of computing → Computation of transforms**; *Mathematical software performance.*

Additional Key Words and Phrases: matching pursuit, signal analysis, GPU, FFT

**ACM Reference Format:**
Piotr T. Różański. 2024. empi: GPU-accelerated matching pursuit with continuous dictionaries. *ACM Trans. Math. Softw.* ?, ?, Article ? (June 2024), 17 pages. https://doi.org/10.1145/not-yet-available

## 1 INTRODUCTION

One of the basic problems in signal analysis is finding a sparse approximation of the input signal by a subset of functions (time-frequency *atoms*) $g_i \in D$ from the pre-defined set (a *dictionary*). Formally, we want to find a sequence of atoms $(g_i)$ and the sequence of coefficients $c_i \in \mathbb{R}$ to minimize the difference

$$\chi = \| x(t) - \sum_{i=1}^{k} c_i \, g_i(t) \| \, . \tag{1}$$

Such a problem may have several variants: we can either fix the number of atoms $k$ and look for the best $k$-sparse approximation, or we can set the threshold $\chi_{\max}$ for the difference and find the minimal $k$ for which a representation with $\chi \leq \chi_{\max}$ exists. Nonetheless, in general, it is an NP-hard problem [3]. However, an approximate solution can be obtained by a greedy algorithm called *matching pursuit*.

### 1.1 Matching pursuit

Matching pursuit is an iterative algorithm for signal decomposition first proposed by Mallat and Zhang in 1993 [14]. Starting with a signal $x(t)$ and a pre-defined dictionary of functions (*atoms*), in each iteration a single element from the dictionary corresponding to the best match with the signal

Author's address: Piotr T. Różański, prozanski@fuw.edu.pl, University of Warsaw, Warszawa, Poland.


0098-3500/2024/6-ART? $15.00
https://doi.org/10.1145/not-yet-available

is found and subtracted from the signal. The residual signal from the given iteration becomes an input signal for the next iteration and the procedure repeats until the stopping criteria are met, e.g.

$$\begin{aligned} x^{(0)} &= x \\ g_i &= \operatorname{argmax}_{g \in D}(g \cdot x^{(i-1)}) \\ c_i &= g_i \cdot x^{(i-1)} \\ x^{(i)} &= x^{(i-1)} - c_i\, g_i\,, \end{aligned} \tag{2}$$

where $\cdot$ denotes the scalar product

$$g \cdot x = \int_{-\infty}^{\infty} g(t)x(t)\,dt \tag{3}$$

and in case of discrete signals (e.g. time series), the integral is replaced with a summation.

The criteria for ending the algorithm can be based either on the pre-selected number of iterations, or on the total residual energy being reduced below a given threshold (e.g. 1% of the initial signal energy). Also, since energies $c_i^2$ of subsequent atoms will form a descending sequence due to inherent properties of matching pursuit [14], the stopping criteria can also be based on the last atom's energy instead.

The most popular approach to constructing a dictionary for matching pursuit is to use oscillating atoms with well-behaving and continuous envelope functions [14]. One of the possible choices are Gabor atoms associated with a Gaussian envelope function, which in addition to having good numerical properties, are also considered to be a good model for transient effects in time series. Moreover, atoms defined this way have a compact representation in the time-frequency space, which opens a way to estimating time-frequency maps of energy density based on matching pursuit [7]. A dictionary can be further supplemented with non-oscillating atoms (e.g. delta-type atoms, where each function has only one non-zero-value) or (co)sine functions, spanning the entire signal.

### 1.2 Continuous vs discrete dictionaries

*Time-frequency dictionaries* introduced in the original formulation of matching pursuit [14] consist of atoms obtained by scaling, translating and modulating a given envelope function (e.g. Gaussian envelope). Assuming a dictionary structured in this manner allows one to define a *parameter space* $\Gamma_0 = \{(s, t_0, f_0)\}$ where $s$ is the atom's scale, $t_0$ is its temporal position and $f_0$ is the modulation frequency. This way, constructing a dictionary $D$ is equivalent to choosing a set of parameters $\Gamma \subset \Gamma_0$ and proceeding with the construction as $D = \{g_\gamma : \gamma \in \Gamma\}$ where $g_\gamma$ is an atom obtained by transforming an envelope function with $\gamma = (s, t_0, f_0)$. (Technical details on performing this transformation will be provided in section 3.4.)

If $\Gamma = \Gamma_0$, the dictionary $D_\infty = \{g_\gamma : \gamma \in \Gamma_0\}$ is itself uncountable and highly redundant. Such dictionaries defined by a continuous parameter space will be called *continuous dictionaries*. On the other hand, any dictionary constructed through some arbitrary finite sampling of the parameter space will be called a *discrete dictionary* throughout this paper. For practical reasons, the prevalent approach in actual signal processing applications has been, up until now, to restrict the calculations to discrete dictionaries.

The structure of a discrete dictionary has been partially specified in the initial paper [14] as a *dyadic* dictionary, characterized by atoms' time scales forming a geometric series. However, the quantities defining exact spacing in time and frequency were left as free parameters. A more precise study on the dictionary structure was presented in [13] and resulted in introducing an *optimal* dictionary, sampling the scale-time-frequency parameter space in a uniform way, in respect to a

well-defined product-related metric. This step reduced the problem of constructing a dictionary to a single degree of freedom, a free parameter $\epsilon$ corresponding to the density of the dictionary.

Even with the optimal dictionary construction, the accuracy of the estimated parameters increase with the size (density) of the dictionary $D$, while at the same time, larger dictionaries (smaller $\epsilon$) increase the computational load significantly. In order to remove this arbitrary parameter affecting both the quality of the resulting decomposition as well as the computation time, a practical method for performing matching pursuit calculations with continuous dictionaries in an effective and mathematically provable way is highly desirable.

Another problem that would be automatically solved as a side effect of introducing continuous dictionaries relates to the statistical bias introduced by MP decomposition in a dictionary with finite and regular sampling of parameters. The solution proposed in 2001 [6] in terms of stochastic dictionaries made most of the numerical optimizations impossible, rendering the algorithm extremely hungry for resources. The algorithm proposed in this paper removes the statistical bias, the necessity of an arbitrary choice of the dictionary's density, and at the same time offers a significant improvement in the speed of computation.

### 1.3 Rationale

The initial motivation for this study comes from the field of electroencephalography (EEG). Matching pursuit (MP) with Gabor dictionaries offers a sparse representation of signals in terms of Gabor functions, parameterized explicitly by phases, time and frequency centers, and widths. Based upon these parameters, one can find correspondence between some of the functions fitted to EEG and structures known from its traditional, visual analysis. This allowed for at least partial unification of the traditional and algorithmic analysis of EEG [4].

In the previous implementations of MP used for EEG analysis up until recently ([7, 8, 13, 21]), accuracy of the estimated parameters increased with the size (density) of the dictionary $D$, with larger dictionaries increasing significantly the computational load. This encouraged several attempts to the optimization of the dictionary structure (c.f. [6, 13, 14]). The implementation presented in this paper can be viewed as a decomposition in a continuous, hence infinitely large, dictionary; its recent application allowed e.g. for an efficient decomposition of ~1000 hours of 19-channel EEG into $\sim 10^7$ atoms in [5].

### 1.4 Outline of the paper

This paper presents a high-performance implementation of matching pursuit based on combining the optimal dictionary construction from [13] with an accurate estimation of the maximum error within a single iteration through the associated *optimality factor* from [14]. The combination of these two elements opens a way to simulating calculations with a continuous dictionary, while preserving the core principle of matching pursuit: a guarantee of choosing the globally best match in each iteration. Apart from removing the dependence of results on $\epsilon$, this step eliminates altogether the statistical bias caused by an arbitrary dictionary structure.

The paper is organized as follows. Section 2 describes the mathematical principles of the proposed method of simulating a continuous dictionary. Section 3 describes the technical details behind the proposed implementation *empi*. Section 4 presents an experimental performance evaluation of *empi*, including a comparison with MPTK [12] as a reference implementation. These three sections are followed by a short summary in section 5, including a discussion on possible further development of this implementation.

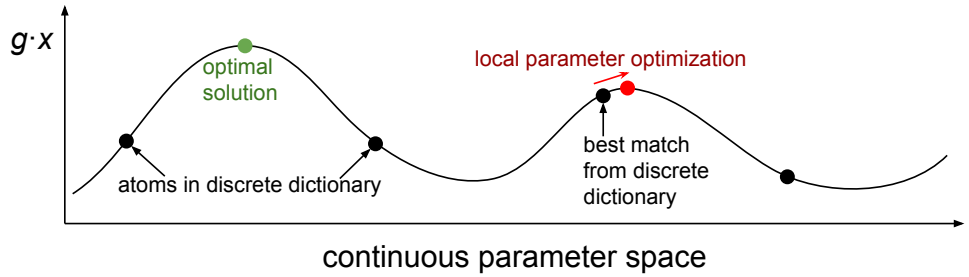


Fig. 1. Example of the configuration in which optimizing the parameters starting only from the best match in a discrete dictionary is not sufficient to find a global maximum.

## 2 METHOD

### 2.1 Simulating a continuous dictionary

The idea of using a discrete dictionary $D$ to calculate a decomposition with a dictionary with continuous parameter space $D_\infty \supset D$ seems to emerge as a natural consequence from Mallat's seminal paper on matching pursuit [14]. For a given signal $x(t)$, the problem can be seen as to find the best match $g_\infty = \operatorname{argmax}_{g\in D_\infty}(g \cdot x)$ starting with a finite number of available scalar products $g \cdot x$ for each $g \in D$.

An intuitive solution would be to find the best match from the dictionary $D$ and then use a local parameter optimization method (gradient or otherwise) to find the best match from the quasi-continuous dictionary. However, this strategy would not guarantee that the atom found this way would actually be the globally best match, as shown schematically on Fig. 1.

To find the actual best match from the continuous dictionary, it is necessary to perform local parameter optimization starting not only from the single best match in $D$, but also from the number of sub-optimal candidates. In order to identify the promising candidates, we have to re-introduce the optimality factor $\alpha < 1$ from [14] as the dictionary-specific value that guarantees that

$$\forall_{g_\infty \in D_\infty} \exists_{g\in D}\, g \cdot x \geq \alpha (g_\infty \cdot x) \tag{4}$$

for all possible input signals $x$. This value of $\alpha$ is directly related to the maximum error in the single iteration that would be made by matching the signal $x$ with the atom $g \in D$ instead of $g_\infty$.

Knowing this value for a given dictionary allows us to perform local parameter optimization only for promising best-match candidates. More specifically, we start by performing local optimization from the best match $g^{(0)} \in D$, obtaining $g_\infty^{(0)} \in D_\infty$ and storing this optimized atom's energy as $c^2_{\max}$. Then we perform the local optimization for other candidates $g \in D$, but only if

$$c^2 = (g \cdot x)^2 \geq \alpha^2 c^2_{\max} \tag{5}$$

and every time the optimization $g \longrightarrow g_\infty$ results in a better match $c^2_\infty > c^2_{\max}$, the latter is updated.

*2.1.1 Assessing the optimality factor.* As we are mainly interested in the neighborhoods of promising best-match candidates (those with a good match to the signal), let's restrict to $x \sim g_\infty$ as the worst-case scenario. In this case

$$\alpha = \min_{g\in D} \min_{g_\infty \in \varepsilon(g)} (g_\infty \cdot g) \tag{6}$$

where $\varepsilon(g)$, the neighborhood of $g$, is defined as the subset of $D_\infty$ for which $g$ is the best approximation:

$$\varepsilon(g) = \{g_\infty \in D_\infty :\ \operatorname{argmax}_{g'\in D}(g_\infty \cdot g') = g\}\,. \tag{7}$$

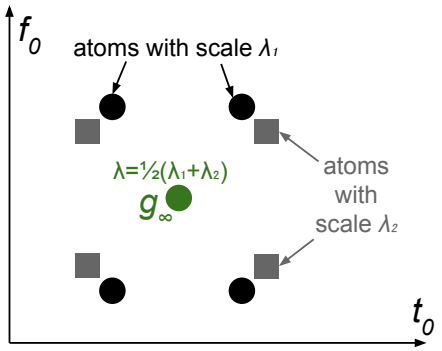


Fig. 2. Atom configuration for which the lowest bound for $\alpha$ is estimated. This plot represents a 2D view of the three-dimensional parameter space in which dictionary functions (atoms) can be identified. Horizontal and vertical axes represent sampling in time and frequency according to the $\Delta_t$ and $\Delta_f$ formulae for optimal dictionaries. The third dimension (scale parameter) is represented through shape: dictionary atoms with scale $\lambda_1$ (in plane) are represented as black circles, while dictionary atoms with neighboring scale $\lambda_2 > \lambda_1$ (out of plane) are represented as gray squares. The function $g_\infty$ not present in the dictionary, associated with the lowest value of $\alpha$ (as in eq. 4) is positioned halfway between planes $\lambda_1$ and $\lambda_2$ and marked with a green dot.

In other words, if we defined the product-based metric (as introduced in [13]) on $D_\infty$, each discrete dictionary $D$ would define the Voronoi diagram partitioning the parameter space and the $\varepsilon(g)$ would be the Voronoi cell associated with the seed $g$. Due to the mismatch of grid dimensions $\Delta_t$ and $\Delta_f$ between neighboring scales differing by $\Delta_\lambda$, the cells $\varepsilon(g)$ may have various shapes. However, a detailed evaluation of possible configurations demonstrates that the lowest bound for $\alpha$ is obtained with a symmetric configuration as shown in Fig. 2.

However, even for such a configuration, the lower bound for $\alpha$ cannot be easily derived analytically due to envelope-related normalization effects [19, 20]. Instead, the numerical analysis can be performed in order to find some practical approximation. The results are shown in Fig. 3. It is shown that the lower bound for $\alpha^2$ for middle frequencies (e.g. half of the Nyquist frequency) is only dependent on $\epsilon^2$, with a good linear fit. On the other hand, the relation between the middle-frequency $\alpha^2_{f_0=0.25}$ and the actual lower bound $\alpha^2$ is dependent only on the product $s \times f_0$, and neither depend on $\epsilon$ nor scale $s$. By combining these two observations, a simplified, yet practical model can be proposed as

$$\alpha^2 \geq (1 - a\epsilon^2)(1 - e^{P(sf_0)}) \tag{8}$$

where $P$ is a polynomial with a small degree (e.g. 1). It should be emphasized that this proposed model is only a rough, practical approximation for calculating a lower bound of $\alpha^2$ and it is not a result of any rigorous analytical derivation.

For Gabor atoms, one possible analytical lower bound, by no means unique, can be obtained with $a = 1.5$ and $P(x) = -1.59x - 2.11$ (Fig. 3b). A similar analysis can be performed for any other envelope function, for example, triangular envelope [20], for which $a = 1.52$ and $P(x) = -0.9x - 1.97$ (Fig. 3d).

## 3 INTRODUCING EMPI

### 3.1 Towards the high-performance implementation

The first version of a high-performance implementation *empi* (**e**nhanced **m**atching **p**ursuit **i**mplementation) was released in 2016 as a more accurate and faster replacement for the previous implementation MP5 [13] developed and used in the University of Warsaw. It has been designed with the same

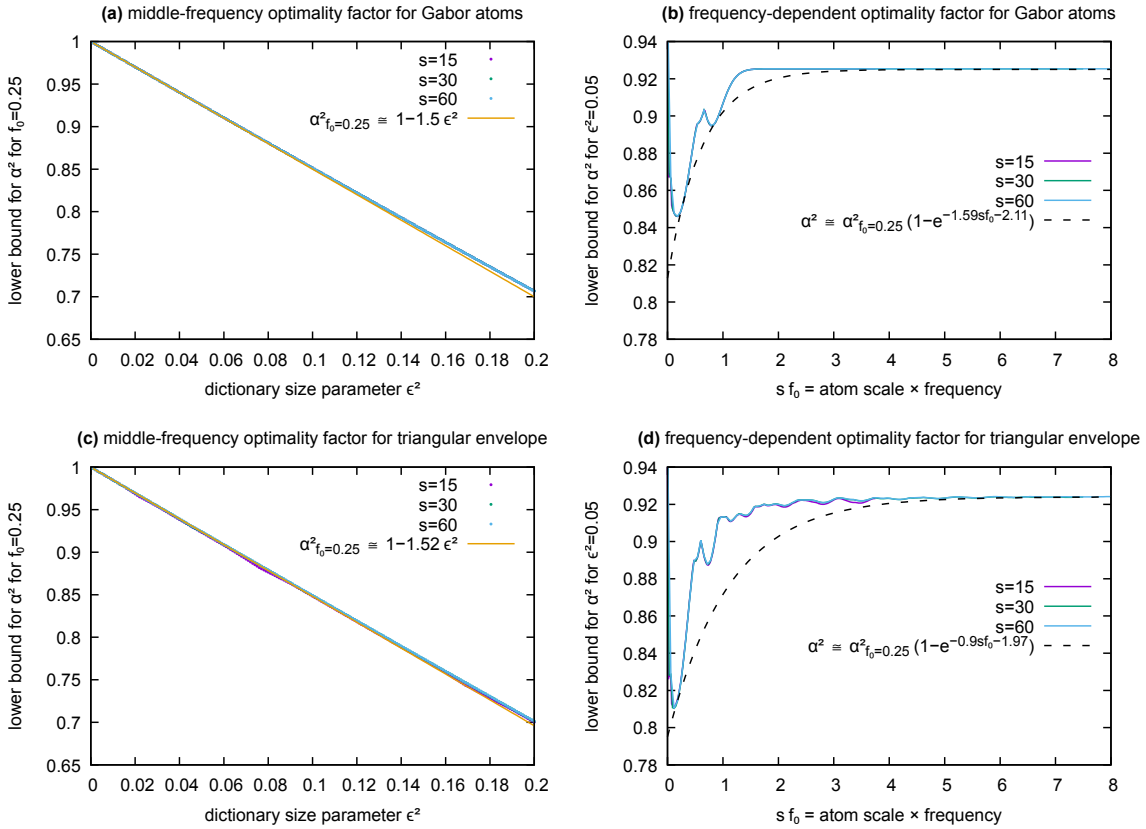


Fig. 3. Numerical study for the lower bound of $\alpha^2$ for two different envelope function: Gaussian (Gabor atoms), **(a-b)**; and triangular envelope function **(c-d)**. The left-hand side images **(a/c)** present the lower bound for $\alpha^2$ restricted to the middle frequency $f_0 = 0.25$ (half of the Nyquist frequency) and show that it depends only on $\epsilon^2$, with a good linear fit. The right hand-side images **(b/d)** present the relation between $\alpha^2_{f=0.25}$ (left-hand side image) and $\alpha^2$ for a number of different scales ($s$ in samples) and show that it depends only on the product $s \times f_0$, as the lines corresponding to various $s$ mostly overlap. A simple analytical approximation for the lower bound of $\alpha^2$ in both cases is also plotted.

input/output specification, so its use with a graphical signal analysis toolkit Svarog [8] required only minor modifications. Few years later, on its way to the first stable version 1.0, the software has been rewritten from scratch with significantly smaller memory footprint, more efficient parallelization and GPU support.

Based on the earlier extension of the optimal dictionary construction to non-Gaussian envelopes [20], this implementation also provides a practical framework for easy introduction of non-Gabor atoms, while preserving the same rigorous treatment. Combining atoms with different envelopes to form a hybrid dictionary is allowed as well.

### 3.2 Related work

To the best of the author's knowledge, no open-source implementation of matching pursuit with continuous dictionaries has been proposed, let alone made available so far. Even for discrete dictionaries, despite the ongoing popularity of matching pursuit and numerous papers dedicated to the mathematical properties of the algorithm, the range of available versatile open-source

implementations is still fairly limited, with most of them lacking proper support for multi-thread and multi-core CPUs. Instead, many published papers have used in-house implementations, usually written in Matlab or Python and not optimized thoroughly.

Apart from MPTK [12] being the most established and widely-tested implementation, one should mention the implementation of [9] and the most recent implementation by [18]. Both of these implementations are restricted to Gabor atoms and offer no support for parallel (multi-threaded) calculations. While the implementation presented in [18] introduces a significant performance gain over MPTK, it relies upon an approximate update strategy using a truncated Gram matrix, which may introduce arbitrary errors into calculations. The errors may be avoided by performing arbitrary buffer resets, but this functionality is available fully only through the Matlab/Octave interface, which limits its applicability for comparative benchmarking. Also, it has not been studied how these errors propagate in case of multi-variate matching pursuit. Therefore, the performance comparison in section 4 will be limited to MPTK as a current de-facto standard implementation of matching pursuit.

### 3.3 General description of *empi*

*empi* is written in a modern (2017) variant of C++. The only external run-time dependency is a fast Fourier transform implementation FFTW [10]. Code from SQLite[1] and CLI11[2] projects is included in the source code itself, along with the appropriate disclaimers and licenses.

The software is open-source (GPL) and can be downloaded from https://github.com/develancer/empi where both source code and Linux binaries are available. Moreover, it can be compiled in various other operating systems, including OS X and Windows, provided CMake is available.

As with most computational tools, *empi* uses the command-line interface. Two arguments are always required: path to the input signal file, and path for the resulting decomposition file to be created. Input signal is always assumed to be raw binary floating-point data. In case of multi-channel signals, data corresponding to separate channels are assumed to be multiplexed. Two output formats for the decomposition file are supported: SQLite (default) and JSON.

In addition to these two parameters, one can use a number of flags and options, controlling various aspects of program behavior. Detailed description of all options is included with the source and binary releases.

There are two modes of CPU parallelization, which can also be used together. First, a number of independent workers can be started, and each worker will process a separate subset of segments and/or channels. Second, each worker can be started with a number of concurrent CPU threads but all threads associated with a given worker will share work on the same signal segment. Either way, in addition to all workers' threads, an additional single thread will be active and responsible for writing the decomposition results from all workers, in proper order, to the output file. The simplified workflow of the program is shown in Fig. 4.

Therefore, if the input signal consists of multiple independent channels or segments to be analyzed separately, both paralelization modes can be used (section 4.2 will provide a performance comparison between this two modes). If the input signal consist of a single segment and a single channel, or is to be processed in a multi-variate mode (where all channels have to be processed simultaneously), thread-based paralelization should be used, as even in case of multiple workers, only one worker would be assigned the entire signal anyway.

If the GPU devices are used in addition to CPU, each worker is started with one additional thread (corresponding to a separate CUDA stream) for each GPU device. The performance gain from

[1] https://www.sqlite.org/
[2] https://github.com/CLIUtils/CLI11

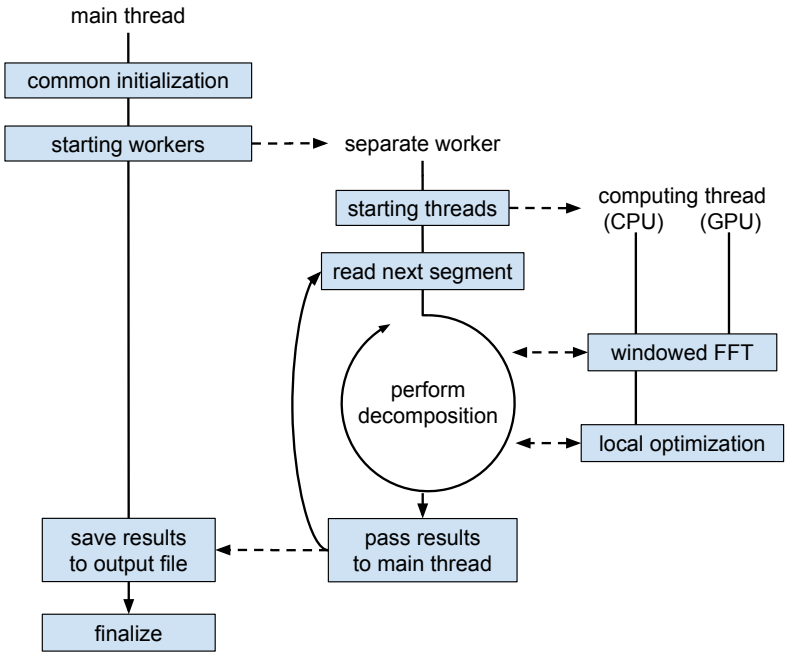


Fig. 4. Simplified workflow of the implementation.

enabling GPU devices, especially those with good double-precision floating point capabilities, is significant and will be evaluated in section 4.

### 3.4 Dictionary construction

The most computationally intensive part of the decomposition process is finding the best match from the discrete dictionary of oscillating atoms (eq. 2). The dictionary is a set of functions constructed by scaling, translating and modulating a pre-defined envelope function $\Phi(t)$, as follows:

$$g_{(s,f_0,t_0)}(t) = K\,\Phi_s(t - t_0)\cos\left(2\pi f_0(t - t_0) + \varphi\right) \tag{9}$$

where $\Phi_s(t) \sim \Phi(\frac{t}{s})$ is a scaled envelope function, $K$ is the normalization factor, resulting from $L^2$-normalization, and all envelope functions (both original and scaled) are normalized as well:

$$\sum_{t=-\infty}^{+\infty} g_{(s,f_0,t_0)}(t)^2 = \sum_{t=-\infty}^{+\infty} \Phi(t)^2 = \sum_{t=-\infty}^{+\infty} \Phi_s(t)^2 = 1\,. \tag{10}$$

The software uses the optimal dictionary construction as introduced in [13], defined by a single parameter $\epsilon$ corresponding to the inverse of the dictionary size. *(Please note this is completely unrelated to a threshold parameter $\epsilon$ introduced in [18].)* According to this construction, the values of scale parameter $s$ form a geometric series, which allows to make a substitution $\lambda = \log s$ and introduce a constant log-scale step $\Delta_\lambda$ instead. For each scale $s$, time ($t_0$) and frequency ($f_0$) values form a rectangular grid with steps of $\Delta_t$ and $\Delta_f$, respectively. Actual values of $\Delta_\lambda$, $\Delta_t$ and $\Delta_f$ can be computed by solving equations

$$A(\Delta_\lambda) = B(s\Delta_f) = C(s^{-1}\Delta_t) = 1 - \epsilon^2 \tag{11}$$

where $A$, $B$ and $C$ are defined as specific integrals of the envelope function $\Phi$. The general formulæ for the these integrals as well as actual results for various envelope functions are presented in [20].

For example, using the Gaussian envelope $\Phi(t) = 2^{1/4}e^{-\pi t^2}$ leads to the following step values for Gabor atoms:

$$\begin{aligned} \Delta_\lambda &= \operatorname{arcosh} \frac{1}{(1-\epsilon^2)^2} \\ \Delta_f &= \frac{1}{s}\sqrt{-\frac{2}{\pi}\log(1-\epsilon^2)} \\ \Delta_t &= s\sqrt{-\frac{2}{\pi}\log(1-\epsilon^2)} \end{aligned} \tag{12}$$

In the current version of *empi*, the available envelope functions are: Gaussian envelope (Gabor atoms) and triangular envelope. However, adding the support of further envelope functions is straightforward and requires only implementing a well-documented class interface named *Family*.

*3.4.1 Boundary conditions.* In order to create a realization of the dictionary based on the construction given above, one needs to set boundary conditions for all three dimensions. The design choice for *empi* is the following:

- For **scale** ($s$ or $\lambda$), the minimum and maximum values should be provided by the user. If not provided, some sensible default values are calculated as described in the documentation.
- For **frequency** ($f_0$), the natural range is from 0 to the Nyquist frequency. The user can provide an alternative maximum value to restrict the dictionary atoms to lower frequencies only.
- For **position** ($t_0$), the range spans the entire signal. An additional flag can be specified to ensure that no atom exceeds the signal range. If the flag is not used (default), *empi* assumes that signal values outside the valid range are zero.

It should be emphasized that even though for extremely narrow scales ($s \to 0$) the notion of atom's frequency ceases to exist, each realization of envelope function is re-normalized after discretization for a given scale $s$ according to the construction described in [20], and therefore does not introduce additional normalization issues.

### 3.5 Scalar product for the best match

In order to find the best match for the input signal, one needs to compute the scalar product between the input signal and each atom in the dictionary. Due to the specific structure of the dictionary, this step can be substantially optimized with the use of FFT. The scalar product between the atom $g(t)$ and the discrete-time signal $x(t)$ can be written as

$$g_{(s,f_0,t_0)} \cdot x = K \sum_{t=-\infty}^{+\infty} \Phi_s(t-t_0)\cos\left(2\pi f_0(t-t_0)+\varphi\right)x(t)\,. \tag{13}$$

On the other hand, the discrete Fourier transform of the input signal, windowed by the scaled envelope function can be written as

$$X_{(s,f_0,t_0)} = \sum_{t=-\infty}^{+\infty} \Phi_s(t-t_0)e^{-2\pi i f_0(t-t_0)}\,x(t)\,. \tag{14}$$

Employing FFT in calculations allows to calculate $X$ for all frequencies $f_0$ for the given pair $(s, t_0)$ through a single transform. However, the relation between equations 13 and 14 is not as straightforward as it seems. The normalization factor $K$ depends on the atom's phase $\varphi$. On the other hand, finding the optimal phase (corresponding to the largest value of the scalar product) requires the exact value of $K$. Therefore, it is not accurate to simply substitute $\arg X$ as the optimal phase.

Instead, one can introduce an intermediate "corrected" value $\tilde{X}$ defined as

$$\tilde{X}_{(s,f_0,t_0)} = \frac{2}{1-|F|^2}\left(X_{(s,f_0,t_0)} - X^*_{(s,f_0,t_0)}F_{(s,f_0)}\right) \tag{15}$$

where $F$ is the Fourier-like sum over the scaled envelope function

$$F_{(s,f_0)} = \sum_{t=-\infty}^{+\infty} \Phi_s(t)^2 e^{-4\pi i f_0 t} \,. \tag{16}$$

For sufficiently large values of $f_0$, $F$ approaches 0, and therefore, $\tilde{X}$ differs from $X$ only by a factor of 2. On the other hand, for very low frequencies, $|F|$ is very close to 1 and for such cases, the implementation uses a slightly different (albeit equivalent) numerical approach.

Based on the corrected value of $\tilde{X}$, one can finally calculate without any loss of precision

- optimal phase $\varphi = \arg \tilde{X}$,
- amplitude = $|\tilde{X}|\,\Phi_s(0)$,
- scalar product $c = g_{(s,f_0,t_0)} \cdot x$ (see eq. 2) and associated atom energy

$$c^2 = \frac{1}{2}\left(|\tilde{X}|^2 + \Re(\tilde{X}^2 F^*)\right) \,. \tag{17}$$

For each scale $s$ and time offset $t_0$, only the highest value of $c^2$ (among all values of $f_0$) is stored in memory, which results in a significantly reduced memory footprint. All stored values of $c^2$ form a two-level heap-like static structure which can be easily updated and in which the highest value is always available.

Also, similarly to the approach used in [12], not all values are re-calculated between iterations, but only those corresponding to parts of the input signal affected by the subtraction of the previous best match. However, the affected scalar products are always updated from scratch instead of using any kind of update formula, as initially proposed in [14] and later used in [13]. There are three main reasons for this design choice:

- using any kind of analytical formula for the scalar product of Gabor atoms would introduce a significant degree of error, especially for the very narrow scales, where the discrete Gabor atoms significantly deviate from their analog counterparts,
- derivation of the exact analytical formulæ for various envelope functions may be very difficult or even impossible, especially if the decomposition is using multiple atom types at once;
- on the other hand, numerical computation of scalar products between all atoms would be a significant bottleneck, both time- and memory-wise.

The pseudo-code of the algorithm is shown in Algorithm 1. Calculating the scalar products in lines 3–5 and 16–17 is optimized with FFT according to equations (14)–(17). Determining the optimal phase for each atom has been omitted for simplicity.

*3.5.1 Low-level use of FFT libraries.* For CPU-based calculations, FFTW [10] is used to calculate all Fourier transforms. Since each iteration of MP requires calculating multiple FFTs, there is no need to parallelize individual transforms. Instead, if multiple threads are associated with the same worker, each of them is assigned a different pool of windowed Fourier transforms to be calculated.

However, for calculations using GPU, a different approach is used. In order to achieve the best performance on massively parallel computing systems, each CUDA kernel call is responsible for calculating multiple FFTs, thanks to the underlying multi-transform routines available in CuFFT [16]. Since CuFFT does not have any built-in support for calculating windowed Fourier transforms (especially if they overlap), this is accomplished with CUDA callbacks and virtual addressing. Also, if at least one GPU is used throughout the computation, all host memory which needs to be read or written in CUDA calls is allocated as page-locked (pinned) memory, accessible to all GPUs.

**Algorithm 1** A simplified outline of the matching pursuit implementation.

**for** each atom scale $s_0$ **do**
  **for** each time shift $t_0$ **do**
    $c^2_{\text{best}}[s_0, t_0] \leftarrow \max_f (g_{(s_0,t_0,f)} \cdot x)^2$
    ▷ find the best match between the signal $x$ and envelope $(s_0, t_0)$
    $f_{\text{best}}[s_0, t_0] \leftarrow \operatorname{argmax}_f (g_{(s_0,t_0,f)} \cdot x)^2$ ▷ and the associated frequency $f$
  **end for**
**end for**
**while** stop conditions are not met **do**
  $s_0, t_0 \leftarrow \operatorname{argmax} c^2_{\text{best}}$
  $f_0 \leftarrow f_{\text{best}}[s_0, t_0]$ ▷ $(s, t_0, f_0)$ defines the best match for this iteration
  $c \leftarrow g_{(s_0,t_0,f)} \cdot x$ ▷ the correct phase-dependent amplitude is computed
  $x \leftarrow x - c\, g_{(s_0,t_0,f_0)}$ ▷ the matched atom is subtracted from signal
  **for** each atom scale $s$ **do**
    **for** each time shift $t$ **do**
      **if** envelopes $(s_0, t_0)$ and $(s, t)$ overlap **then**
        $c^2_{\text{best}}[s, t] \leftarrow \max_f (g_{(s,t,f)} \cdot x)^2$
        $f_{\text{best}}[s, t] \leftarrow \operatorname{argmax}_f (g_{(s,t,f)} \cdot x)^2$
      **end if**
    **end for**
  **end for**
**end while**

In each iteration, all transforms which need to be computed are sorted according to their length and placed in the doubly ended queue. This way, CPU threads consume short transforms first and leave the long transforms to GPU threads, for which the efficiency of CuFFT is far more superior.

*3.5.2 Local parameter optimization.* Local optimization of the parameters is performed with a Nelder-Mead method [15]. More specifically, *empi* uses the C++ port[3] of the original O'Neill code [17] with subsequent remarks by Chambers & Ertel [2], Benyon [1] and Hill [11]. The only change to this code added by the author is using simplex size instead of terminating variance as the optimalization's stopping criteria (which can be further adjusted with command-line options).

The function to be maximized is the atom's energy $c^2$ (eq. 5), and the optimization is performed in the space of $(\lambda, f_0, t_0)$ with the parameters scaled using local values of $\Delta_\lambda$, $\Delta_f$ and $\Delta_t$ so the simplex size is well-defined and isotropic. Results of local optimization are stored in a result cache as a map $g \mapsto g_\infty$, and each time the part of the signal overlapping any atom $g$ changes, the cache entry $g_\infty$ associated with that atom is erased.

## 4 EXPERIMENTAL EVALUATION

All benchmark calculations have been performed on a 23-channel 1-hour long EEG recording with 128 Hz sampling rate, extracted from the middle of an overnight sleep recording. In all tests, the dictionary consisted of Gabor atoms only, and the scale parameter ($s$) spanned the range between 0.1 and 10 seconds.

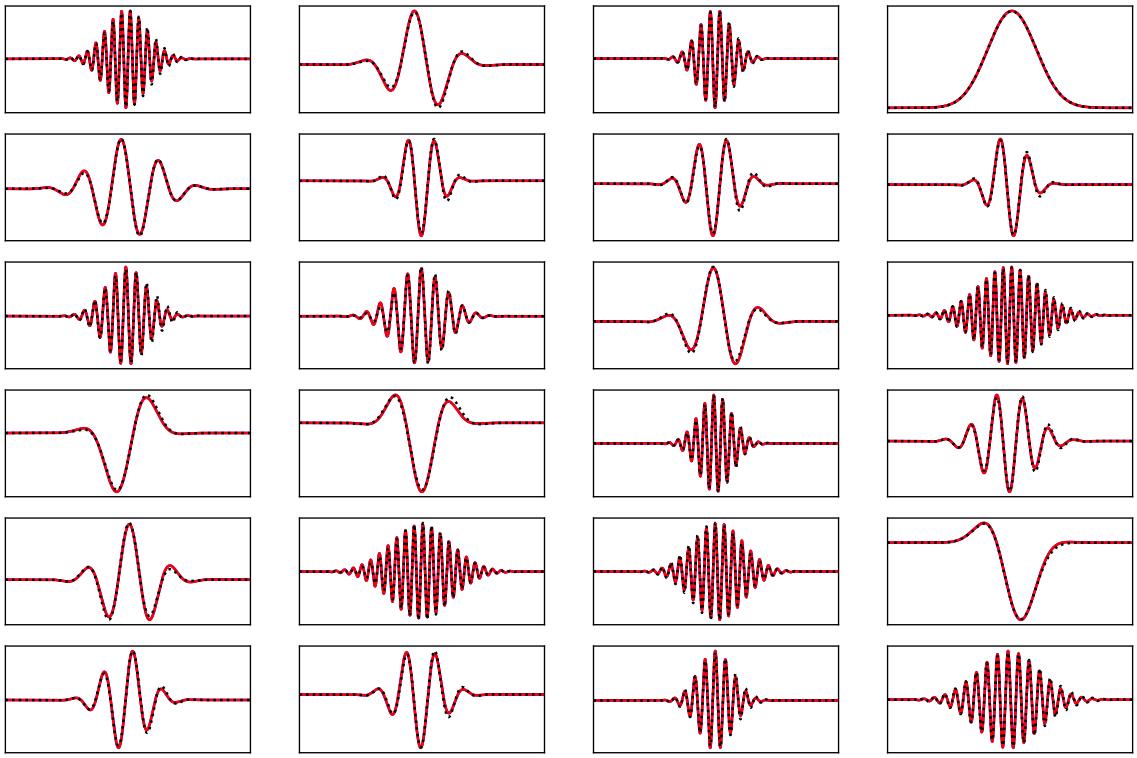

Fig. 5. Results of a series of single-iteration runs of matching pursuit algorithm on artificial signals composed of individual Gabor atoms. Signal reconstructions obtained with MPTK and *empi* represented by solid lines fully overlap, while the black dotted line represents the actual signal.

### 4.1 Comparison with MPTK

In order to test the numerical correctness of *empi* as well as quantitatively measure its performance, MPTK [12] has been chosen as a reference, as being an established and widely used implementation of matching pursuit. The first step was to establish a correspondence between parameters sets of both implementations in order to calculate decompositions using the same dictionary structure. This was accomplished by adapting *empi* to support

- a flag `--dictionary-output`, exporting the dictionary structure as a MPTK-compatible XML file, and
- an additional switch `--full-atoms-in-signal` restricting the atoms in the dictionary to fully overlap with the signal's time range.

Decomposition results of artificial signal segments, simulated as single Gabor functions with randomly selected parameters, are shown in Fig. 5. Lines representing functions fitted in the first iterations by MPTK and *empi* overlap completely, confirming bit-perfect agreement between these two implementations obtained by a suitable choice of corresponding sets of parameters. The black dotted line represents the actual signal, which may not fully coincide with the reconstruction since the simulations used a discrete dictionary with $\epsilon^2 = 0.01$ (same for both implementations) while the Gabor atoms' parameters were chosen randomly from continuous distributions.

[3]https://github.com/develancer/nelder-mead

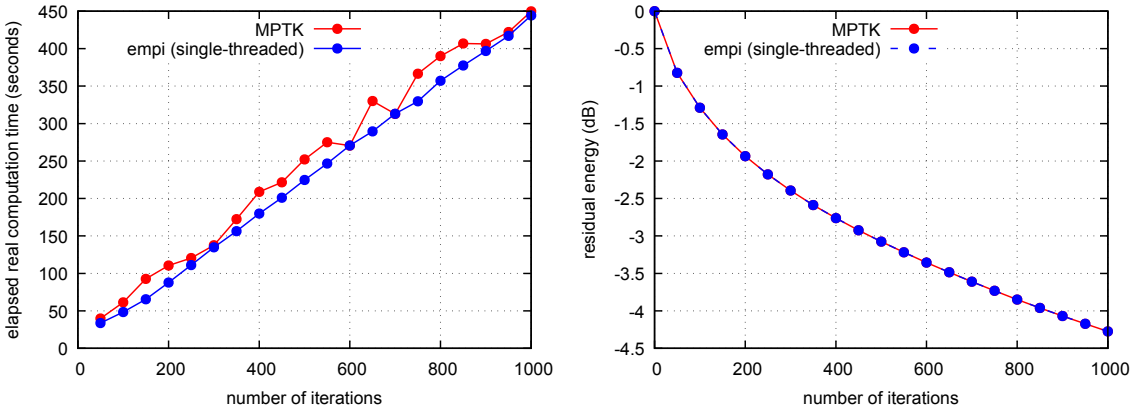


Fig. 6. Performance comparison of MPTK and *empi* for a discrete dictionary, using a single CPU thread. Left-hand side plot represents a real (wall-time) computation time, and the right-hand side plot represents a residual energy after a given number of iterations. Each data point represents a separate computation on the same hour-long single-channel EEG recording. The benchmark calculations were performed on a customer-grade workstation with Intel Core i5-8400 (2.80 GHz) CPU.

To compare the performance of these two implementations on a real signal, a series of decompositions has been performed on a single channel extracted from an hour-long EEG recording. Similarly as above, the same dictionary with $\epsilon^2 = 0.01$ and over 650 million Gabor atoms has been used for both implementations. Additionally, to make a fair comparison, *empi* was restricted to one CPU thread, as MPTK does not support multi-threading. The results provided in Fig. 6 demonstrate that even in a single-threaded case, *empi* provides a similar (but less varied) performance compared to MPTK, while the resulting decomposition and the corresponding residual signal energy is exactly the same for both implementations. The gain from *empi*'s multi-threading (both CPU and GPU) will be evaluated in section 4.2.

The last comparison, using the same single-channel signal, was performed using a finer discrete dictionary with $\epsilon^2 = 0.001$. For each implementation, 100 iterations were performed and the resulting decomposition was once again found to be identical. Despite using a significantly larger dictionary consisting of more than 32 billion atoms, both implementations demonstrated a very modest memory footprint of around 200 MB, which confirms the applicability of *empi* to analyze longer signals without any need of splitting the data into separate segments. The total wall-clock computation time was 1313.65 s for MPTK and 1002.61 s for a single-threaded *empi* run.

## 4.2 Multi-threading benchmarks

To evaluate performance in multi-threaded and hybrid CPU+GPU usage, benchmark calculations were performed on a high-performance computing system consisting of a 16-core AMD EPYC 7302 processor and two NVIDIA A100 GPUs. The EEG signal was either analyzed as a single hour-long segment, or split into 60 segments (one minute each). For each segment, the decomposition was performed until the residual was found to be below 1% (−20 dB) of the signal's initial energy.

*4.2.1 Single-channel decomposition.* The first benchmark used a discrete dictionary with $\epsilon^2 = 0.01$. Only a single channel from the recording (corresponding to Cz electrode) was used in this test. Results using various numbers of CPU workers and threads, both with and without GPU, are presented in Fig. 7. As long as the signal is split into multiple segments **(a)**, both parallelization schemes work similarly well, with a slight edge towards the scheme based on independent workers.

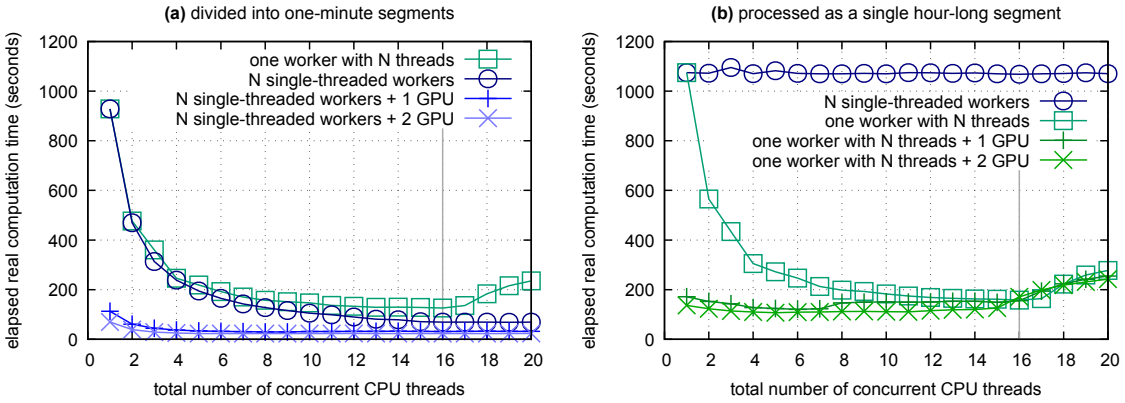


Fig. 7. Real (wall-time) elapsed computation time for a discrete dictionary decomposition of a hour-long single-channel EEG recording for different parallelization strategies and GPU usage. The vertical line represents the system's CPU capacity of 16 cores.

| | SMP | MMP1 | MMP3 |
|---|---|---|---|
| CPU only (16 workers) | 1240.11 | 7916.84 | 2781.94 |
| CPU + 1 GPU | 560.06 | 2262.44 | 1141.13 |
| CPU + 2 GPU | 363.66 | 1477.13 | 794.60 |

Table 1. Elapsed real computation time (in seconds) of multivariate matching pursuit decomposition of an hour-long 23-channel EEG recording.

However, if only a single segment is available **(b)**, multiple workers cannot be used efficiently and the thread-based approach has to be used. In both cases, enabling one or two GPUs in addition to CPU processing power offers a significant (up to 10×) reduction in computation time.

*4.2.2 Multi-channel decomposition.* Multi-channel decomposition was computed using all 23 channels, with the same dictionary as above ($\epsilon^2 = 0.01$). Three variants were tested: SMP (where each channel is processed independently), MMP1 corresponding multi-channel decomposition with constant phase and MMP3 corresponding to multi-channel decomposition with variable phase. Detailed description of these three decomposition modes can be found in the literature, e.g. [13]. The signal was divided into 60 segments (one minute each) and the computation was performed here only in one configuration, employing 16 independent workers. For each decomposition mode, 0, 1, or 2 GPUs were enabled in addition to the CPU processing power. The results shown in Table 1 demonstrate a significant speed-up gained from utilizing GPUs.

### 4.3 Continuous dictionary

Calculations for simulated continuous dictionary were performed for a single channel only, similarly to section 4.2.1. Two available parameter optimization modes were tested: local (where parameter optimization is performed only locally around the best match) and global (according to the systematic approach described in section 2.1) as well as no-optimization (discrete dictionary) mode, each of them for a wide range of $\epsilon^2$ values. The aim was to measure not only the total computation time, but also the quality of decomposition in each case, measured as the amount of residual energy after

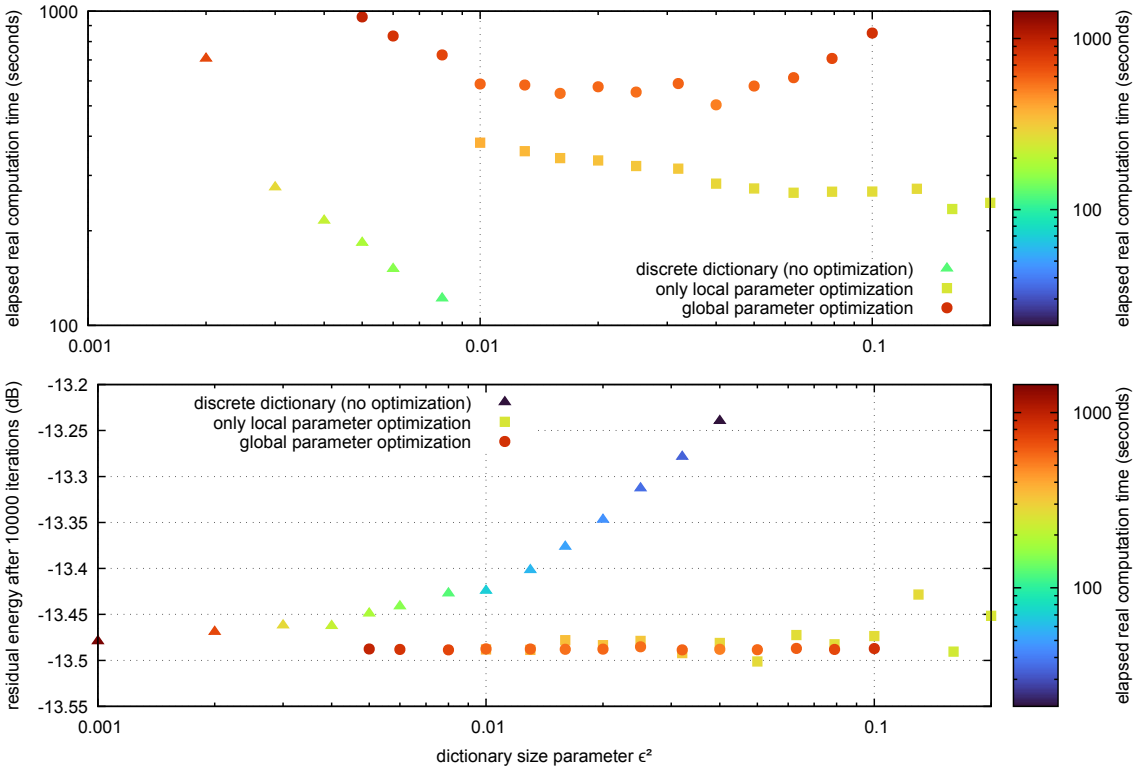


Fig. 8. Benchmark results for decomposition of an hour-long single-channel EEG recording in 10000 iterations. Shapes represent strategies for parameter optimization in each iteration: triangles—plain discrete dictionary (no optimization), rectangles—local search around the best candidate, and circles—global optimization proposed in this paper (section 2.1). Horizontal axis—density of the dictionary, decreasing with $\epsilon^2$ (section 3.4). Upper plot shows total wall-clock time for each computation run, marked also with colors on both plots. Lower plot—residual signal energy. Simulating a continuous dictionary offers stable results independent on the initial dictionary size, with computation times increasing for small and large dictionaries.

10000 iterations. The results shown in Fig. 8 correspond to the same EEG signal as analyzed in the previous section.[4]

If no optimization is used, the performance shows a monotonic dependency between the size of the dictionary and the computation time (triangles in the top half of Fig. 8). This is expected, as a larger dictionary always requires more computing power to calculate FFTs and related scalar products for each atom. Enlarging the dictionary also provides a decrease in the residual energy (the bottom half of Fig. 8), as the atoms in subsequent iterations provide a better fit to the signal. Together, the results constitute a trade-off between computation time and the quality of the decomposition.

For the local optimization mode, the results (squares in Fig. 8) are somewhat different. The computation time is significantly longer than in the previous case due to additional time needed to locally optimize the parameters of each and every atom. Due to this additional step, the quality of the decomposition is significantly better. However, the variation in the overall quality is also significant and corresponds to the fact that the local optimization mode is not enough to fully simulate a continuous dictionary, as there is no guarantee of selecting the globally best match in

[4]However, it should be noted that tests were also performed on a random (white noise) signal and demonstrated a qualitatively similar behavior as described below, but with expectedly higher residual signal energy for the same number of iterations.

each iteration. Moreover, since smaller dictionaries decrease the probability of accurately sampling the global optimum, the variation on the vertical scale grows as we decrease the dictionary size (right-most yellow-green squares on the bottom half of Fig. 8).

However, for the global optimization mode (red circles int Fig. 8), the computation time increases both for very fine (e.g. $\epsilon^2 = 0.005$) as well as for very coarse (e.g. $\epsilon^2 = 0.079$) dictionaries. For very small values of $\epsilon^2$, the increase in computation time can be attributed to a large number of atoms, similarly as in two previous cases. However, for very large $\epsilon^2$, the computation time increases as well, and this can be attributed to a large number of candidate atoms for which the local decomposition must be performed. As the $\alpha^2$ (see section 2.1) decreases, the number of atoms fulfilling eq. 5 grows as well, contributing to an increase in the overall computation time.

On the other hand, the quality of the decomposition in the global optimization mode does not depend on the selected value of $\epsilon^2$. Therefore, for the global optimization mode there is always an optimal value for $\epsilon^2$ providing the same results in the shortest time. For the analyzed system this value was found to be around $\epsilon^2 = 0.04$, but it is expected to vary across different machines.

It should be emphasized that although the difference in residual energy between various modes of operation may not seem significant, and one can argue that increasing the number of iterations may make the residual energy arbitrarily small, the provided quantities are also indicators of the accuracy of the determined parameters. In many applications, including but not limited to biomedical engineering, e.g. analysis of sleep spindles (or in general, EEG sleep profiles) [21], obtaining the precise values of parameters ($s$, $t_0$, $f_0$) for each atom in the decomposition is crucial, as any mismatch in those parameters may lead to misclassification. Moreover, properly simulating the continuous dictionary removes the ambiguity associated with the $\epsilon^2$ parameter and eliminates the impact of an arbitrary dictionary structure on the resulting decomposition.

## 5 CONCLUSIONS AND FUTURE WORK

This paper introduced a mathematical framework for matching pursuit algorithm with continuous dictionaries and presented an open-source implementation *empi* with multi-threading and multi-GPU capabilities, performing exact calculations including, but not limited to, Gabor atoms. The proposed global optimization strategy based on evaluating mathematical properties of the optimal dictionary structure was shown to provide consistent results simulating a continuous dictionary. Multiple benchmarks were provided, demonstrating a significant performance gain resulting from multi-threading and/or GPU usage.

The work is (as always) far from done and there is still potential for optimization, in both GPU- and CPU-related parts of code. Especially, the implementation of finding the correct common phase for atoms in the MMP1 variant could probably be optimized, as it seems from the results in Table 1 that it is a bottleneck for multivariate decomposition. The local optimization routine, currently using a customized version of the Nelder-Mead algorithm, could benefit from introducing a different approach, but this will require a systematic evaluation of all possible solutions. Last but not least, one can notice in Fig. 8 that even in the global optimization mode, there is still some deviation between results for various $\epsilon^2$. This effect may be attributed to a finite accuracy of local optimization, but some further research is still needed to exclude additional factors.

## ACKNOWLEDGEMENTS

The author would like to thank his PhD supervisors, prof. Piotr Durka and prof. Krzysztof Stencel from the University of Warsaw, for enormous patience and many valuable suggestions, as well as dr Marian Dovgialo for testing the software and providing constructive feedback. The benchmark calculations in sections 4.2 and 4.3 have been performed on a high-performance computing cluster at the Faculty of Physics, Astronomy and Informatics, Nicolaus Copernicus University in Toruń.

This research was partly supported by a grant from the Polish National Science Centre UMO-2018/31/B/ST7/01888.

## REFERENCES


[1] P. R. Benyon. 1976. Remark AS R15: Function Minimization Using a Simplex Procedure. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 25, 1 (1976), 97–97. http://www.jstor.org/stable/2346539

[2] John M. Chambers and J. E. Ertel. 1974. Remark AS R11: A Remark on Algorithm AS 47 "Function Minimization using a Simplex Procedure". *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 23, 2 (1974), 250–251. http://www.jstor.org/stable/2347015

[3] G. Davis, Stéphane G. Mallat, and M. Avellaneda. 1997. Adaptive greedy approximations. *Constructive Approximation* 13, 1 (01 Mar 1997), 57–98. https://doi.org/10.1007/BF02678430

[4] Piotr J. Durka. 2007. *Matching Pursuit and Unification in EEG analysis.* Artech House, Norwood, MA. ISBN 978-1-58053-304-1.

[5] Piotr J. Durka, Marian Dovgialo, Anna Duszyk-Bogorodzka, and Piotr Biegański. 2024. Two-Stage Atomic Decomposition of Multichannel EEG and the Previously Undetectable Sleep Spindles. *Sensors* 24, 3 (2024), 14. https://www.mdpi.com/1424-8220/24/3/842

[6] Piotr J. Durka, Dobiesław Ircha, and Katarzyna Blinowska. 2001. Stochastic time-frequency dictionaries for matching pursuit. *IEEE Transactions on Signal Processing* 49, 3 (2001), 507–510. https://doi.org/10.1109/78.905866

[7] Piotr J. Durka, Dobiesław Ircha, Christa Neuper, and Gert Pfurtscheller. 2001. Time-frequency microstructure of event-related electro-encephalogram desynchronisation and synchronisation. *Medical and Biological Engineering and Computing* 39 (2001), 315–321. https://doi.org/10.1007/BF02345286

[8] Piotr J. Durka, Urszula Malinowska, Magdalena Zieleniewska, Christian O'Reilly, Piotr T. Różański, and Jarosław Żygierewicz. 2015. Spindles in Svarog: framework and software for parametrization of EEG transients. *Frontiers in Human Neuroscience* 9 (2015), 258. https://doi.org/10.3389/fnhum.2015.00258

[9] Sebastian E. Ferrando, Lawrence A. Kolasa, and Natasha Kovačević. 2002. Algorithm 820: A Flexible Implementation of Matching Pursuit for Gabor Functions on the Interval. *ACM Trans. Math. Softw.* 28, 3 (sep 2002), 337–353. https://doi.org/10.1145/569147.569151

[10] Matteo Frigo and Steven G. Johnson. 2005. The Design and Implementation of FFTW3. *Proc. IEEE* 93, 2 (2005), 216–231. Special issue on "Program Generation, Optimization, and Platform Adaptation".

[11] I. D. Hill. 1978. Remark AS R28: A Remark on Algorithm AS 47: Function Minimization Using a Simplex Procedure. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 27, 3 (1978), 380–382. http://www.jstor.org/stable/2347187

[12] Sacha Krstulovic and Rémi Gribonval. 2006. MPTK: Matching Pursuit made Tractable. In *Proc. Int. Conf. Acoust. Speech Signal Process. (ICASSP'06)*, Vol. 3. Toulouse, France, III–496 – III–499.

[13] Rafał Kuś, Piotr T. Różański, and Piotr J. Durka. 2013. Multivariate matching pursuit in optimal Gabor dictionaries: theory and software with interface for EEG/MEG via Svarog. *BioMedical Engineering OnLine* 12, 1 (23 Sep 2013), 94. https://doi.org/10.1186/1475-925X-12-94

[14] Stéphane G. Mallat and Zhifeng Zhang. 1993. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing* 41, 12 (1993), 3397–3415. https://doi.org/10.1109/78.258082

[15] J. A. Nelder and R. Mead. 1965. A Simplex Method for Function Minimization. *Comput. J.* 7, 4 (01 1965), 308–313. https://doi.org/10.1093/comjnl/7.4.308

[16] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. 2020. CUDA, release: 10.2.89. https://developer.nvidia.com/cuda-toolkit

[17] R. O'Neill. 1971. Algorithm AS 47: Function Minimization Using a Simplex Procedure. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 20, 3 (1971), 338–345. http://www.jstor.org/stable/2346772

[18] Zdeněk Průša, Nicki Holighaus, and Peter Balazs. 2021. Fast Matching Pursuit with Multi-Gabor Dictionaries. *ACM Trans. Math. Softw.* 47, 3, Article 24 (jun 2021), 20 pages. https://doi.org/10.1145/3447958

[19] Piotr T. Różański. 2018. Normalization Effects in Matching Pursuit Algorithm with Gabor Dictionaries. *Journal of Applied Computer Science* 26, 2 (2018), 187–199.

[20] Piotr T. Różański. 2020. Effects of envelope and dictionary structure on the performance of matching pursuit. *IET Signal Processing* 14, 2 (2020), 89–96. https://doi.org/10.1049/iet-spr.2019.0246

[21] Magdalena Zieleniewska, Anna Duszyk, Piotr T. Różański, Marcin Pietrzak, Marta Bogotko, and Piotr J. Durka. 2019. Parametric Description of EEG Profiles for Assessment of Sleep Architecture in Disorders of Consciousness. *International Journal of Neural Systems* 29, 03 (2019), 1850049. https://doi.org/10.1142/S0129065718500491 PMID: 30642201.