RENSx アルゴリズム解説

RENSx algorithm: a versatile source localization algorithm for various types of biomagnetic imaging

株式会社　シグナルアナリシス　関原謙介

1　はじめに

RENSx アルゴリズムは脳磁界計測を始めとして，脊髄磁場計測，末梢神経磁場計測あるいは心磁界計測など，生体磁場計測のいろいろな分野において有用な信号源推定アルゴリズムである．RENSx アルゴリズムはスパースベイズラーニング（sparse Bayes learning, SBL）をベースにしたアルゴリズムであるが，生体磁気計測データからの信号源再構成問題に対応するため，SBL を土台としていくつかの工夫・改良が施されている．一般的な SBL アルゴリズムについては，ベイズ信号処理 [1] の第 5 章やシグナルアナリシス社技術資料 [2] に詳しい解説があるので，このノートは，このアルゴリズムの特に信号源再構成問題に特化した部分を解説する．このノートにおける説明は，ほぼ Electromagnetic Brain Imaging (EBI)[3] の第 4 章の内容に沿ったものである．

2　種々の定義とデータモデル

2.1　データベクトルとソースベクトル

m 番目のセンサーの時刻 t における出力を $y_m(t)$ で表し，センサーアレイ全体の出力を列ベクトル

$$\boldsymbol{y}(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_M(t) \end{bmatrix} \tag{1}$$

で表す．ここで，M はセンサーアレイに含まれるセンサー数である．3 次元空間における位置をベクトル $\boldsymbol{r} = (x, y, z)$ で表す．位置 $\boldsymbol{r}$ に計測の対象となっている信号を発している信号源が存在すると仮定する．信号源は時間的に変動する 3 次元ベクトルであり，位置 $\boldsymbol{r}$ における時刻 t の信号源（ソース）ベクトルを $\boldsymbol{s}(\boldsymbol{r})$ と表し，

$$\boldsymbol{s}(\boldsymbol{r}, t) = \begin{bmatrix} s_x(\boldsymbol{r}, t) \\ s_y(\boldsymbol{r}, t) \\ s_z(\boldsymbol{r}, t) \end{bmatrix} \tag{2}$$

とする．ここで $s_x(\boldsymbol{r}, t)$, $s_y(\boldsymbol{r}, t)$, $s_z(\boldsymbol{r}, t)$ はソースベクトルの x, y, z 成分である．

2.2 センサーリードフィールド（センサー感度ベクトル）

次にセンサーリードフィールドを定義する．位置 $\boldsymbol{r}$ に単位強度の信号源が存在すると仮定し，この信号源が，x 方向を向いている場合のこの単位強度の信号源による m 番目のセンサーの出力を $l_m^x(\boldsymbol{r})$ とする．同様に，この信号源が y 方向を向いている場合の出力を $l_m^y(\boldsymbol{r})$, z 方向を向いている場合の出力を $l_m^z(\boldsymbol{r})$ とする．すると，3 次元の行ベクトル $\boldsymbol{l}_m(\boldsymbol{r})$,

$$\boldsymbol{l}_m(\boldsymbol{r}) = [l_m^x(\boldsymbol{r}), l_m^y(\boldsymbol{r}), l_m^z(\boldsymbol{r})] \tag{3}$$

が m 番目のセンサーの位置 $\boldsymbol{r}$ における感度を表す．この $\boldsymbol{l}_m(\boldsymbol{r})$ を用いて，$M \times 3$ の行列を

$$\boldsymbol{L}(\boldsymbol{r}) = \begin{bmatrix} \boldsymbol{l}_1(\boldsymbol{r}) \\ \vdots \\ \boldsymbol{l}_M(\boldsymbol{r}) \end{bmatrix} \tag{4}$$

と定義すれば，この $\boldsymbol{L}(\boldsymbol{r})$ がセンサーアレイ全体の位置 $\boldsymbol{r}$ における感度を表す．この行列 $\boldsymbol{L}(\boldsymbol{r})$ はリードフィールド行列と呼ばれる．このリードフィールド行列を用いて，観測ベクトル $\boldsymbol{y}$ と空間に分布した信号源ベクトル $\boldsymbol{s}(\boldsymbol{r})$ の関係は

$$\boldsymbol{y} = \int_\Omega \boldsymbol{L}(\boldsymbol{r})\boldsymbol{s}(\boldsymbol{r})\, d\boldsymbol{r} \tag{5}$$

と表現できる． ここで，右辺の積分は信号源が存在する可能性のある 3 次元の領域 Ω について行う．この Ω を信号源空間 (source space) と呼ぶ.

2.3 離散ソースモデル

式 (5) に示される積分方程式を，信号源空間をボクセル分割により離散化し，線型方程式に変換する．各ボクセルの座標を $\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_N$ とする．ここでボクセル上に存在する離散的な信号源で信号源分布が近似できるとして，信号源分布をデルタ関数を用いて

$$\boldsymbol{s}(\boldsymbol{r}, t) = \sum_{j=1}^{N} \boldsymbol{s}(\boldsymbol{r}_j, t)\delta(\boldsymbol{r} - \boldsymbol{r}_j)$$

と表せば，これを式 (5) に代入して

$$\boldsymbol{y}(t) = \sum_{j=1}^{N} \boldsymbol{L}(\boldsymbol{r}_j)\boldsymbol{s}(\boldsymbol{r}_j, t) \tag{6}$$

を得る．ここで以下に定義される $M \times 3N$ 行列

$$\boldsymbol{F} = [\boldsymbol{L}(\boldsymbol{r}_1), \boldsymbol{L}(\boldsymbol{r}_2), \ldots, \boldsymbol{L}(\boldsymbol{r}_N)] \tag{7}$$

を導入する．この $\boldsymbol{F}$ はボクセルリードフィールド行列と呼ばれる．さらに，$3N \times 1$ の列ベクトル $\boldsymbol{x}(t)$ を

$$\boldsymbol{x}(t) = \begin{bmatrix} \boldsymbol{s}(\boldsymbol{r}_1, t) \\ \boldsymbol{s}(\boldsymbol{r}_2, t) \\ \vdots \\ \boldsymbol{s}(\boldsymbol{r}_N, t) \end{bmatrix} \tag{8}$$

と定義する．この $\boldsymbol{x}(t)$ はすべてのボクセル位置の信号源ベクトルを含む列ベクトルであり，式 (2) で定義されたソースベクトルと区別するためボクセルソースベクトルあるいはボクセルベクトルと呼ぶ．すると，式 (6) は

$$\boldsymbol{y}(t) = [\boldsymbol{L}(\boldsymbol{r}_1), \boldsymbol{L}(\boldsymbol{r}_2), \ldots, \boldsymbol{L}(\boldsymbol{r}_N)] \begin{bmatrix} \boldsymbol{s}(\boldsymbol{r}_1, t) \\ \boldsymbol{s}(\boldsymbol{r}_2, t) \\ \vdots \\ \boldsymbol{s}(\boldsymbol{r}_N, t) \end{bmatrix} = \boldsymbol{F}\boldsymbol{x}(t) \tag{9}$$

と表すことができる．さらに，観測結果には，信号成分に加法的にノイズ $\boldsymbol{\varepsilon}$ が重畳すると仮定すれば，観測データのモデルは

$$\boldsymbol{y}(t) = \boldsymbol{F}\boldsymbol{x}(t) + \boldsymbol{\varepsilon} \tag{10}$$

となる．ノイズベクトル $\boldsymbol{\varepsilon}$ は M 次元の列ベクトルで $\boldsymbol{\varepsilon} = [\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_M]^T$ であり，j 番目の要素 ε_j は j 番目の観測データ $y_j(t)$ に重畳するノイズを表す．式 (10) が観測データを表す線形離散モデルである．

データが時刻 $t_1, t_2, \ldots, t_K$ で計測される時系列データであるとする．表記法を簡単にするため時刻 t_k で計測されたデータベクトルを $\boldsymbol{y}_k$（$\boldsymbol{y}(t_k) = \boldsymbol{y}_k$）で表し，時刻 t_k のソースベクトルを $\boldsymbol{x}_k$（$\boldsymbol{x}(t_k) = \boldsymbol{x}_k$）と表す．すると，式 (10) は

$$\boldsymbol{y}_k = \boldsymbol{F}\boldsymbol{x}_k + \boldsymbol{\varepsilon} \tag{11}$$

と表される．時系列データの全体 $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_K$ を $\boldsymbol{y}$ で表す．(つまり，$\boldsymbol{y}$ と書いてあれば $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_K$ を意味する．) 同様に，$\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_K$ を $\boldsymbol{x}$ と表す．後の議論のため行列 $\boldsymbol{F}$ の j 番目の列ベクトルを $\boldsymbol{f}_j$ で表す．すなわち，$\boldsymbol{f}_j$ を用いると

$$\boldsymbol{F} = [\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_{3N}]$$

である．

3　スパースベイズ学習

3.1　確率モデル

式 (11) で表されたのモデルのもとで，観測データ $\boldsymbol{y}$ から未知な信号源分布 $\boldsymbol{x}$ を推定する問題を考える．ノイズの確率分布として平均ゼロ，共分散行列 $\boldsymbol{\Sigma_\varepsilon}$ の多次元正規分布

$$p(\boldsymbol{\varepsilon}) = \mathcal{N}(\boldsymbol{\varepsilon}|\boldsymbol{0}, \boldsymbol{\Sigma_\varepsilon}) \tag{12}$$

を仮定する．式 (11) のデータモデルと式 (12) のノイズの確率分布からデータ $\boldsymbol{y}_k$ の条件付き確率分布 $p(\boldsymbol{y}_k|\boldsymbol{x}_k)$ が

$$p(\boldsymbol{y}_k|\boldsymbol{x}_k) = \mathcal{N}(\boldsymbol{y}_k|\boldsymbol{F}\boldsymbol{x}_k, \boldsymbol{\Sigma_\varepsilon}) \tag{13}$$

と求まる．

ベイズ学習ではこれから推定する未知量 $\boldsymbol{x}$ に対する事前分布を導入する．スパースベイズ学習では事前分布として，要素 $x_j(t)$ に対して分散が ν_j で，要素間で独立な正規分布：

$$p(x_j(t)) = \mathcal{N}(x_j(t)|0, \nu_j) \tag{14}$$

を仮定する．ベクトル確率変数 $\boldsymbol{x}_k$（$\boldsymbol{x}_k \in \mathbb{R}^{3N\times 1}$）に対する確率分布は

$$p(\boldsymbol{x}_k) = \mathcal{N}(\boldsymbol{x}_k|\boldsymbol{0}, \boldsymbol{\Upsilon}) \tag{15}$$

となる．この事前分布の共分散行列 $\boldsymbol{\Upsilon}$（$\boldsymbol{\Upsilon} \in \mathbb{R}^{3N\times 3N}$）はベクトル $\boldsymbol{\nu}$（$\boldsymbol{\nu} \in \mathbb{R}^{3N\times 1}$）を $\boldsymbol{\nu} = [\nu_1, \nu_2, \ldots, \nu_{3N}]^T$ と定義して，

$$\boldsymbol{\Upsilon} = \mathrm{diag}(\boldsymbol{\nu}) \tag{16}$$

と表される対角行列となる．つまり $\boldsymbol{\Upsilon}$ はこの $\boldsymbol{\nu}$ の要素を対角成分として持つ対角行列である．

3.2　事後確率分布の導出

ボクセルソースベクトル $\boldsymbol{x}_k$ のもっとも確からしい推定解は，事後分布 $p(\boldsymbol{x}_k|\boldsymbol{y}_k)$ を最大とする $\boldsymbol{x}_k$ として求めることができる（MAP（maximum a posteriori）推定解）．事前分布（式 (15)）と尤度（式 (13)）が正規分布の場合，事後分布 $p(\boldsymbol{x}_k|\boldsymbol{y}_k)$ も正規分布で与えられる．したがって，事後分布を平均 $\bar{\boldsymbol{x}}_k$，精度行列（共分散行列の逆行列）$\boldsymbol{\Gamma}$ の正規分布

$$p(\boldsymbol{x}_k|\boldsymbol{y}_k) = \mathcal{N}(\boldsymbol{x}_k|\bar{\boldsymbol{x}}_k, \boldsymbol{\Gamma}^{-1}) \tag{17}$$

とすれば，精度行列と平均は

$$\boldsymbol{\Gamma} = \boldsymbol{\Upsilon}^{-1} + \boldsymbol{F}^T\boldsymbol{\Sigma}_\varepsilon^{-1}\boldsymbol{F} \tag{18}$$

$$\bar{\boldsymbol{x}}_k = \boldsymbol{\Gamma}^{-1}\boldsymbol{F}^T\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1}\boldsymbol{y}_k \tag{19}$$

として求まる[1]．したがって，事後分布 $p(\boldsymbol{x}_k|\boldsymbol{y}_k)$ を最大とする解，すなわち，未知量 $\boldsymbol{x}_k$ の MAP 推定解は，正規分布では期待値で確率最大となるのであるから，式 (19) で与えられる $\bar{\boldsymbol{x}}_k$ である．

式 (19) を等価な別の表現に変換する．逆行列公式（ベイズ信号処理 (A.51)）を用いて，

$$\bar{\boldsymbol{x}}_k = (\boldsymbol{\Upsilon}^{-1} + \boldsymbol{F}^T\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1}\boldsymbol{F})^{-1}\boldsymbol{F}^T\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1}\boldsymbol{y}_k = \boldsymbol{\Upsilon}\boldsymbol{F}^T(\boldsymbol{F}\boldsymbol{\Upsilon}\boldsymbol{F}^T + \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}})^{-1}\boldsymbol{y}_k = \boldsymbol{\Upsilon}\boldsymbol{F}^T\boldsymbol{\Sigma}_y^{-1}\boldsymbol{y}_k \tag{20}$$

が成り立つ．ここで，

$$\boldsymbol{\Sigma}_y = \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{F}\boldsymbol{\Upsilon}\boldsymbol{F}^T \tag{21}$$

であり，この $\boldsymbol{\Sigma}_y$ はモデルデータ共分散行列と呼ぶ．さらに，後の議論のために式 (20) を重み行列を用いて

$$\bar{\boldsymbol{x}}_k = \boldsymbol{\Omega}\boldsymbol{y}_k \tag{22}$$

と書く．つまり，事後平均 $\bar{\boldsymbol{x}}_k$ は観測データ $\boldsymbol{y}_k$ の線形推定解として表され，$\bar{\boldsymbol{x}}_k$ を推定する重み $\boldsymbol{\Omega}$ は

$$\boldsymbol{\Omega} = \boldsymbol{\Upsilon}\boldsymbol{F}^T\boldsymbol{\Sigma}_y^{-1} \tag{23}$$

で与えられる．

以上の議論から明らかであるが，未知量 $\boldsymbol{x}_k$ のベイズ推定解（MAP 推定解）$\bar{\boldsymbol{x}}_k$ を計算するにはハイパーパラメータ $\boldsymbol{\Upsilon}$（すなわちその対角成分である $\boldsymbol{\nu}$）とノイズ共分散行列 $\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}$ が必要である．RENSx アルゴリズムではこれらを観測データ $\boldsymbol{y}$ から推定する．以下にまず，ハイパーパラメータ $\boldsymbol{\Upsilon}$ の推定ついて説明する．

3.3 ハイパーパラメータ $\boldsymbol{\nu}$ の学習

3.3.1 周辺尤度の最大化による $\boldsymbol{\nu}$ の学習

A. 周辺尤度の導出

ハイパーパラメータ $\boldsymbol{\Upsilon}$，すなわちその対角成分 $\boldsymbol{\nu}$ の最適推定解 $\widehat{\boldsymbol{\mu}}$ は時系列データの周辺尤度

$$\log p(\boldsymbol{y}|\boldsymbol{\nu}) = \log p(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_K|\boldsymbol{\nu}) = \sum_{k=1}^{K} \log p(\boldsymbol{y}_k|\boldsymbol{\nu})$$

を最大とする $\boldsymbol{\nu}$ として求めることができる．ここで，$\log p(\boldsymbol{y}_k|\boldsymbol{\nu})$ は k 番目のタイムサンプルにおける観測データ $\boldsymbol{y}_k$ に対する周辺尤度である．この周辺尤度 $p(\boldsymbol{y}_k|\boldsymbol{\nu})$ を求めるにはいくつかの方法がある．最も直截的な方法は

$$p(\boldsymbol{y}_k|\boldsymbol{\nu}) = \int p(\boldsymbol{y}_k, \boldsymbol{x}_k|\boldsymbol{\nu})d\boldsymbol{x}_k = \int p(\boldsymbol{y}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{\nu})d\boldsymbol{x}_k \tag{24}$$

から求めるものである．式 (24) において $p(\boldsymbol{y}_k|\boldsymbol{x}_k)$ は式 (13) で与えられ，$p(\boldsymbol{x}_k|\boldsymbol{\nu})$ はで与えられるのでこれらを代入し，右辺の積分を計算すれば $p(\boldsymbol{y}_k|\boldsymbol{\nu})$ を求めることができる．この正攻法による周辺

[1] ベイズ信号処理の 3.4 節を参照のこと．

尤度の計算は「ベイズ信号処理」5.3 節に詳しく説明してある．ただし，この積分の実行にはかなりな計算を必要とする．この積分を計算する方法よりも若干計算の楽な別の方法について「信号処理のための線形代数入門」[4] の 8.2.3 項に説明がある．ここでは，周辺尤度 $p(\boldsymbol{y}_k|\boldsymbol{\nu})$ のさらに簡単な計算法を示す．

ここでは，

$$\log p(\boldsymbol{y}_k|\boldsymbol{\nu}) = -\frac{1}{2}\log|\boldsymbol{\Sigma}_y| - \frac{1}{2}\left[(\boldsymbol{y}_k - \boldsymbol{F}\bar{\boldsymbol{x}}_k)^T\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1}(\boldsymbol{y}_k - \boldsymbol{F}\bar{\boldsymbol{x}}_k) + \bar{\boldsymbol{x}}^T\boldsymbol{\Upsilon}^{-1}\bar{\boldsymbol{x}}\right] \tag{25}$$

と求まることを証明する．式 (12) に示すように

$$\boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{\varepsilon}|\boldsymbol{0}, \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}})$$

を仮定している，また，式 (11) と (15) が成り立つので，

$$\boldsymbol{F}\boldsymbol{x}_k \sim \mathcal{N}(\boldsymbol{F}\boldsymbol{x}_k|\boldsymbol{0}, \boldsymbol{F}\boldsymbol{\Upsilon}\boldsymbol{F}^T) \tag{26}$$

が成り立つ．確率変数 $\boldsymbol{y}_k$ は独立な正規分布に従う 2 つの確率変数 $\boldsymbol{F}\boldsymbol{x}_k$ と $\boldsymbol{\varepsilon}$ の和で表される．独立な正規分布の和はやはり正規分布であり，

$$\boldsymbol{y}_k = \boldsymbol{F}\boldsymbol{x}_k + \boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{y}_k|\boldsymbol{0}, \boldsymbol{F}\boldsymbol{\Upsilon}\boldsymbol{F}^T + \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}) = \mathcal{N}(\boldsymbol{y}_k|\boldsymbol{0}, \boldsymbol{\Sigma}_y) \tag{27}$$

である[2]．したがって，

$$\log p(\boldsymbol{y}_k) = -\frac{1}{2}\log|\boldsymbol{\Sigma}_y| - \frac{1}{2}\boldsymbol{y}_k^T\boldsymbol{\Sigma}_y^{-1}\boldsymbol{y}_k \tag{28}$$

を得る．さらに，

$$\boldsymbol{y}_k^T\boldsymbol{\Sigma}_y^{-1}\boldsymbol{y}_k = (\boldsymbol{y}_k - \boldsymbol{F}\bar{\boldsymbol{x}}_k)^T\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1}(\boldsymbol{y}_k - \boldsymbol{F}\bar{\boldsymbol{x}}_k) + \bar{\boldsymbol{x}}_k^T\boldsymbol{\Upsilon}^{-1}\bar{\boldsymbol{x}}_k \tag{29}$$

の関係[3]を用いれば結局，時系列データに対する周辺尤度として

$$\log p(\boldsymbol{y}|\boldsymbol{\nu}) = \sum_{k=1}^{K}\log p(\boldsymbol{y}_k|\boldsymbol{\nu}) = -\frac{K}{2}\log|\boldsymbol{\Sigma}_y| - \frac{1}{2}\sum_{k=1}^{K}\left[(\boldsymbol{y}_k - \boldsymbol{F}\bar{\boldsymbol{x}}_k)^T\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1}(\boldsymbol{y}_k - \boldsymbol{F}\bar{\boldsymbol{x}}_k) + \bar{\boldsymbol{x}}_k^T\boldsymbol{\Upsilon}^{-1}\bar{\boldsymbol{x}}_k\right] \tag{30}$$

を得る．ここで，$\boldsymbol{\Sigma}_y$ は式 (21) に示すモデルデータ共分散行列である．

B. ハイパーパラメータ更新式の導出

ハイパーパラメータ $\boldsymbol{\nu}$ の最適な推定解 $\widehat{\boldsymbol{\nu}}$ は式 (30) において

$$\frac{\partial}{\partial\boldsymbol{\nu}}\log p(\boldsymbol{y}|\boldsymbol{\nu}) = 0$$

とおいて求まる $\boldsymbol{\nu}$ である．この微分計算は「ベイズ信号処理」5.4 節で詳しく述べられていおり，事後共分散行列 $\boldsymbol{\Sigma}$ を $\boldsymbol{\Sigma} = \boldsymbol{\Gamma}^{-1}$ と定義して，$\boldsymbol{\Sigma}$ の $\{j,j\}$ 要素を $\Sigma_{j,j}$ と書けば，

$$\frac{\partial}{\partial\nu_j}\log p(\boldsymbol{y}|\boldsymbol{\nu}) = -K\Sigma_{j,j} + K\nu_j - \sum_{k=1}^{K}\bar{x}_j(t_k) \tag{31}$$

[2]「統計的信号処理」の 2.2 節参照.

[3]証明は「信号処理のための線形代数入門」の問題 8.1 の解答を参照のこと

となる．ここで，$\bar{x}_j(t_k)$ はベクトル $\boldsymbol{x}_k$ の j 番目の要素である．したがって，上式の右辺をゼロと置くことにより ν_j の推定解として，

$$\widehat{\nu}_j = \Sigma_{j,j} + \frac{1}{K}\sum_{k=1}^{K} \bar{x}_j(t_k) \tag{32}$$

を得る．(式 (32) は EM アルゴリズムを用いて導出した ν_j の推定解と同じものとなっている．詳しくは「ベイズ信号処理」の第 5 章参照.）式 (32) は推定が劣決定の場合（特にその度合いが激しい場合，すなわち $M << N$ の場合）に収束が遅いことが知られている．次節では収束の早い $\boldsymbol{\nu}$ の学習について述べる．

3.3.2 凸関数の性質を用いた $\boldsymbol{\nu}$ の学習

A. 代替コスト関数の導出

式 (32) よりも収束の早い更新式を凸関数の性質を用いて導出する．(「ベイズ信号処理」5.5 節にここで用いる手法についての解説がある.）コスト関数を式 (30) の周辺尤度を用いて

$$\mathcal{F}(\boldsymbol{\nu}) = -(2/K)\log p(\boldsymbol{y}|\boldsymbol{\nu}) = \log|\boldsymbol{\Sigma}_y| + \frac{1}{K}\sum_{k=1}^{K}\left[(\boldsymbol{y}_k - \boldsymbol{F}\bar{\boldsymbol{x}}_k)^T\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1}(\boldsymbol{y}_k - \boldsymbol{F}\bar{\boldsymbol{x}}_k) + \bar{\boldsymbol{x}}_k^T\boldsymbol{\Upsilon}^{-1}\bar{\boldsymbol{x}}_k\right] \tag{33}$$

と定義する．この $\mathcal{F}(\boldsymbol{\nu})$ を最小とする $\boldsymbol{\nu}$ が $\boldsymbol{\nu}$ のベストな推定解である．この最小化に際して，$\log|\boldsymbol{\Sigma}_y|$ は変数 $\nu_1, \nu_2, \ldots, \nu_{3N}$ に関して上に凸な関数であるので，任意の $\boldsymbol{\nu}$ に対して，z_0 をある定数として

$$\boldsymbol{z}^T\boldsymbol{\nu} - z_0 \geq \log|\boldsymbol{\Sigma}_y| \tag{34}$$

を満たすベクトル $\boldsymbol{z} = [z_1, \ldots, z_{3N}]^T$ が必ず存在するという事実を用いる．すなわち，代替コスト関数を

$$\widetilde{\mathcal{F}}(\boldsymbol{\nu}, \boldsymbol{z}) = \boldsymbol{z}^T\boldsymbol{\nu} - z_0 + \frac{1}{K}\sum_{k=1}^{K}\left[(\boldsymbol{y}_k - \boldsymbol{F}\bar{\boldsymbol{x}}_k)^T\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1}(\boldsymbol{y}_k - \boldsymbol{F}\bar{\boldsymbol{x}}_k) + \bar{\boldsymbol{x}}_k^T\boldsymbol{\Upsilon}^{-1}\bar{\boldsymbol{x}}_k\right] \tag{35}$$

とすれば，

$$\widetilde{\mathcal{F}}(\boldsymbol{\nu}, \boldsymbol{z}) \geq \mathcal{F}(\boldsymbol{\nu}) \tag{36}$$

が成り立つので，代替コスト関数 $\widetilde{\mathcal{F}}(\boldsymbol{\nu}, \boldsymbol{z})$ は真のコスト関数 $\mathcal{F}(\boldsymbol{\nu})$ の上限を形成する．すなわち，代替コスト関数 $\widetilde{\mathcal{F}}(\boldsymbol{\nu}, \boldsymbol{z})$ を補助変数 $\boldsymbol{z}$ と共に変数 $\boldsymbol{\nu}$ で最小化すれば，$\widetilde{\mathcal{F}}(\boldsymbol{\nu}, \boldsymbol{z})$ を最小化する $\boldsymbol{\nu}$ は真のコスト関数をも最小化する．したがって，$\boldsymbol{\nu}$ の最適推定値 $\widehat{\boldsymbol{\nu}}$ は代替コスト関数 $\widetilde{\mathcal{F}}(\boldsymbol{\nu}, \boldsymbol{z})$ を最小とする $\boldsymbol{\nu}$ として求めることができる．

B. ハイパーパラメータ $\boldsymbol{\nu}$ の推定式の導出

実際に $\widehat{\boldsymbol{\nu}}$ は,

$$\widehat{\boldsymbol{\nu}} = \underset{\boldsymbol{\nu}}{\operatorname{argmin}} \widetilde{\mathcal{F}}(\boldsymbol{\nu}, \boldsymbol{z}) = \underset{\boldsymbol{\nu}}{\operatorname{argmin}} \left[\boldsymbol{z}^T \boldsymbol{\nu} + \frac{1}{K} \sum_{k=1}^{K} \bar{\boldsymbol{x}}_k^T \boldsymbol{\Upsilon}^{-1} \bar{\boldsymbol{x}}_k \right]$$
$$= \underset{\boldsymbol{\nu}}{\operatorname{argmin}} \sum_{j=1}^{N} \left[z_j \nu_j + \frac{\frac{1}{K} \sum_{k=1}^{K} \bar{x}_j^2(t_k)}{\nu_j} \right] \quad (37)$$

として求める．再右辺を ν_j に関し微分してゼロと置けば,

$$\widehat{\nu}_j = \sqrt{\frac{\frac{1}{K} \sum_{k=1}^{K} \bar{x}_j^2(t_k)}{z_j}} \quad (38)$$

を得る．この式に示す $\widehat{\nu}_j$ が ν_j の推定解である．しかし上式には，補助変数 z_j が入っている．補助変数 z_j の推定値は（「ベイズ信号処理」5.5.2 項を参照のこと.）

$$\widehat{z}_j = \frac{\partial}{\partial \mu_j} \log |\boldsymbol{\Sigma}_y| \quad (39)$$

として求まる．ここで,

$$\frac{\partial}{\partial \mu_j} \log |\boldsymbol{\Sigma}_y| = \operatorname{tr} \left[\boldsymbol{\Sigma}_y^{-1} \frac{\partial}{\partial \mu_j} \boldsymbol{\Sigma}_y \right] \quad (40)$$

であり，式 (21) から

$$\frac{\partial}{\partial \mu_j} \boldsymbol{\Sigma}_y = \frac{\partial}{\partial \mu_j} \boldsymbol{F} \boldsymbol{\Upsilon} \boldsymbol{F}^T = \boldsymbol{f}_j \boldsymbol{f}_j^T$$

であるので,

$$\widehat{z}_j = \operatorname{tr} \left[\boldsymbol{\Sigma}_y^{-1} \boldsymbol{f}_j \boldsymbol{f}_j^T \right] = \boldsymbol{f}_j^T \boldsymbol{\Sigma}_y^{-1} \boldsymbol{f}_j \quad (41)$$

を得る．式 (38) および (41) が $\boldsymbol{\nu}$ の更新式である．式 (38) に (41) を代入してして

$$\widehat{\nu}_j = \sqrt{\frac{\frac{1}{K} \sum_{k=1}^{K} \bar{x}_j^2(t_k)}{\boldsymbol{f}_j^T \boldsymbol{\Sigma}_y^{-1} \boldsymbol{f}_j}} \quad (42)$$

として $\boldsymbol{\nu}$ の更新式を表すこともできる．

3.4 ハイパーパラメータ・タイイング

前節で述べたアルゴリズムには問題がある．ボクセルソースベクトル $\boldsymbol{x}$ の j 番目の要素が $j = 3\times n+1$ と表せれば，これは n 番目のボクセルの x 成分の値であり，この場合，$j = 3\times n+2$ および $j = 3\times n+3$ ば，n 番目のボクセルの y および z 成分の値である．このようにソースベクトルのインデックス j は異なる 2 つのインデックス (ボクセルのインデックス n と同じボクセルにおけるソースベクトルの 3 つの成分を表すインデックス) の和で表されるが，これまで述べてきたスパースベイズアルゴリズムはこの違いを考慮せず同等に扱うため，信号源の成分間でスパース制約が働いてしまい，信号源の向

きの推定に（その結果として信号源強度の推定にも）誤差を生じてしまう．したがって，このアルゴリズムによる再構成結果は誤差を含む．この実例は付録 B（Biomag2014 の発表ポスター）を参照されたい．

この問題に対する解決策は同じボクセル由来のハイパーパラメータ ν_{3n+1}，ν_{3n+2}，ν_{3n+3} を結び付けて同じ 1 つの値を持つようにすることである．すなわち，それぞれの更新値 $\widehat{\nu}_{3n+1}$，$\widehat{\nu}_{3n+2}$，$\widehat{\nu}_{3n+3}$ を用いて新しい更新値 $\widehat{\mu}_{3n+m}$（$m = 1, 2, 3$）を

$$\widehat{\mu}_{3n+m} = \sqrt{\frac{1}{3}\sum_{i=1}^{3}\widehat{\nu}_{3n+i}^{2}} \tag{43}$$

とすることにより，n 番目のボクセルのハイパーパラメータの値を 3 つの成分に関して同じ値にするのである．これを Hyperparameter Tying[4] と呼ぶ．

ハイパーパラメータを結合する，すなわち同じ値にしてしまうことによりパラメータ間のスパース制約をコントロールできる．すなわち，タイイングしたパラメータの間ではスパースな制約は消失するが，タイイングしていないパラメータの間ではスパースな制約が保たれる．ここで議論している式 (43) を用いた場合には，各ボクセルのソースベクトルの成分間ではスパース制約は消失するが，ボクセル間ではスパース制約はなお有効である[5]．Hyperparameter Tying がスパース制約に与える影響についてのコスト関数を使った議論は付録 B あるいは Electromagnetic Brain Imaging の 4.9 節に説明がある．また，補助変数 $\widehat{z}_j$ についても同様に

$$\widehat{z}_{3n+m}^{new} = \frac{1}{3}\sum_{i=1}^{3}\widehat{z}_{3n+i} \quad (m = 1, 2, 3) \tag{44}$$

とすることによりタイイングを行う．

このハイパーパラメータ・タイイングの有効性に関するコスト関数を用いた議論と実験結果を発表したポスター（Biomag2014,Halifax, Canada）を付録 B に示す．ポスター 6 ページ目に，2 次元データの場合におけるハイパーパラメータ・タイイングを行った場合と行わない場合のコスト関数の導出を示す．また，ポスター 7 ページ目にコスト関数のプロットを示す．このプロットから，ハイパーパラメータ・タイイングを行なわない場合にはコスト関数が L_0-あるいは L_1-ノルムに類似のスパースな解を生ずるタイプの形状をしているが，ハイパーパラメータ・タイイングによってこの形状が L_2-ノルムのコスト関数に類似したスパースな解を生じないタイプの形状に変化することが示される．

8 ページから 9 ページにかけて，ハイパーパラメータ・タイイングの有効性を示すコンピュータシミュレーションとその結果を示す．9 ページの結果は，(a) 事前分布の共分散行列を対角行列とする方法，(b) 事前分布の共分散行列を対角行列として，ハイパーパラメータ・タイイングを行う方法，(c) 事前分布の共分散行列を対角行列とする方法（A 節に述べる方法）を比較し，(a) では再構成結果に大きな誤差を含むが，(b) であれば誤差の無い再構成を得ることができる．また，(c) でも誤差の無い再構成が可能であるが，計算時間が 12 倍（方法 (b) と比べて）かかることが述べられている．

[4] ハイパーパラメータの結合とでも訳すべきか?

[5] ちなみにベイズミニマムノルム法はこの極端な場合である．この方法では全ボクセルのハイパーパラメータを結合し，1 つの値で表していると考えることができる．したがって，ベイズミニマムノルム法ではスパースな解は生じない．なお，ベイズミニマムノルム法については Electromagnetic Brain Imaging の 2.10 節，あるいは技術ノート（参考文献 [5]）を参照のこと．

3.5 ハイパーパラメータのスムージング

SBL アルゴリズムは理論的にはシングルタイムサンプルのデータに対しても有効なアルゴリズムである．しかし，実際にはある程度の数のタイムサンプルを必要とする．この，ある程度の数のタイムサンプルの必要性は式 (38) において明らかである．式 (38) において，ハイパーパラメータ ν_j の更新式は j 番目のボクセルにおける再構成パワーの測定タイムタイムサンプルでの平均値

$$\frac{1}{K}\sum_{k=1}^{K}\bar{x}_j^2(t_k) \tag{45}$$

を用いて計算される．個々のタイムサンプルにおける値 $\bar{x}_j^2(t_k)$ は観測データに混入するノイズの影響を受けるので，ある瞬時値ではなく，式 (45) に示すようにタイムタイムサンプルでの平均値を用いて ν_j の更新を行うことにより，観測データに混入するノイズに対しより安定な更新を行うことができる．

脳磁界計測以外の生体磁気では，少ないタイムサンプルでの信号源再構成を行わなければならない場合も多い．例えば，脊髄や神経磁場の計測では信号源が高速で移動するため，通常はシングルタイムサンプルで信号源再構成を行う．ハイパーパラメータのスムージングは，このような少数タイムサンプルを用いた再構成において少しでも安定した再構成を行うことを目的とする．

ハイパーパラメータスムージングは，式 (43) を用いて計算したハイパーパラメータの更新値 $\widehat{\boldsymbol{\mu}}$ $(= [\widehat{\mu}_1, \ldots, \widehat{\mu}_{3N}]$ を，あるスムージング重み関数 $\boldsymbol{\Theta}$ を用いて，

$$\widehat{\boldsymbol{\mu}}^S = \boldsymbol{\Theta}\widehat{\boldsymbol{\mu}} \tag{46}$$

としてスムージングを行うことにより求めた $\widehat{\boldsymbol{\mu}}^S$ を最終的な更新値として用いる．ハイパーパラメータスムージングを行うことにより，少数タイムサンプルの場合においてもノイズの影響を含む更新値を周囲のボクセルにおける重み付き平均を計算することにより，更新値に含まれるノイズの影響を低減し，アルゴリズムの安定性の向上が期待できる．

4 ノイズの学習

ノイズ共分散行列をハイパーパラメータとみなせば，周辺尤度関数の最大化から，ノイズ共分散行列も推定することができる．ノイズ共分散行列の推定をノイズ学習（noise learning）と呼ぶ．ノイズ共分散行列の最適解は，周辺尤度を最大とするノイズ共分散行列として，すなわち

$$\frac{\partial}{\partial \boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}}\log p(\boldsymbol{y}|\boldsymbol{\nu}, \boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}) = \frac{\partial}{\partial \boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}}\sum_{k=1}^{K}\log p(\boldsymbol{y}_k|\boldsymbol{\nu}, \boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}) = 0 \tag{47}$$

を満たす $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}$ として求めることができる．ただし，この計算のためにはコスト関数を直接最大化するよりも EM アルゴリズムを用いて平均データ尤度を最大化する方が計算が楽である．

ノイズ共分散行列 $\boldsymbol{\Sigma_\varepsilon}$ ではなく精度行列 $\boldsymbol{\Lambda}$（$= \boldsymbol{\Sigma_\varepsilon}^{-1}$）を用いて平均データ尤度を表すと，

$$\Theta(\boldsymbol{\Lambda}) = \sum_{k=1}^{K} E\left[\log p(\boldsymbol{y}_k|\boldsymbol{x}_k, \boldsymbol{\Lambda})\right] + \sum_{k=1}^{K} E\left[\log p(\boldsymbol{x}_k|\boldsymbol{\Upsilon})\right] \tag{48}$$

ここで，$E[\cdot]$ は事後分布の期待値を表す．$\boldsymbol{\Lambda}$ の更新値は

$$\frac{\partial}{\partial \boldsymbol{\Lambda}}\Theta(\boldsymbol{\Lambda}) = \mathbf{0}$$

を満たす $\boldsymbol{\Lambda}$ である．式 (48) において右辺第 2 項は $\boldsymbol{\Lambda}$ を含まずこの計算には関係しない．したがって，

$$\frac{\partial}{\partial \boldsymbol{\Lambda}}\Theta(\boldsymbol{\Lambda}) = \sum_{k=1}^{K} E\left[\frac{\partial}{\partial \boldsymbol{\Lambda}}\log p(\boldsymbol{y}_k|\boldsymbol{x}_k, \boldsymbol{\Lambda})\right] \tag{49}$$

であり，

$$\log p(\boldsymbol{y}_k|\boldsymbol{x}_k, \boldsymbol{\Lambda}) = \frac{1}{2}\log|\boldsymbol{\Lambda}| - \frac{1}{2}(\boldsymbol{y}_k - \boldsymbol{F}\boldsymbol{x}_k)^T\boldsymbol{\Lambda}(\boldsymbol{y}_k - \boldsymbol{F}\boldsymbol{x}_k) \tag{50}$$

である．式 (50) では計算に関係のない定数は書くのを省略した．ここで，

$$\frac{\partial}{\partial \boldsymbol{\Lambda}}\log p(\boldsymbol{y}_k|\boldsymbol{x}_k, \boldsymbol{\Lambda}) = \frac{1}{2}\boldsymbol{\Lambda}^{-1} - \frac{1}{2}(\boldsymbol{y}_k - \boldsymbol{F}\boldsymbol{x}_k)(\boldsymbol{y}_k - \boldsymbol{F}\boldsymbol{x}_k)^T \tag{51}$$

であるので，

$$\frac{\partial}{\partial \boldsymbol{\Lambda}}\Theta(\boldsymbol{\Lambda}) = \frac{K}{2}\boldsymbol{\Lambda}^{-1} - \frac{1}{2}\sum_{k=1}^{K} E\left[(\boldsymbol{y}_k - \boldsymbol{F}\boldsymbol{x}_k)(\boldsymbol{y}_k - \boldsymbol{F}\boldsymbol{x}_k)^T\right] \tag{52}$$

を得る．さらに，

$$E\left[(\boldsymbol{y}_k - \boldsymbol{F}\boldsymbol{x}_k)(\boldsymbol{y}_k - \boldsymbol{F}\boldsymbol{x}_k)^T\right] = (\boldsymbol{y}_k - \boldsymbol{F}\bar{\boldsymbol{x}}_k)(\boldsymbol{y}_k - \boldsymbol{F}\bar{\boldsymbol{x}}_k)^T + \boldsymbol{F}\boldsymbol{\Gamma}^{-1}\boldsymbol{F}^T \tag{53}$$

であるので，結局，$\boldsymbol{\Lambda}$ すなわち $\boldsymbol{\Sigma_\varepsilon}$ の更新式として

$$\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{\varepsilon}} = \widehat{\boldsymbol{\Lambda}}^{-1} = \frac{1}{K}\sum_{k=1}^{K}(\boldsymbol{y}_k - \boldsymbol{F}\bar{\boldsymbol{x}}_k)(\boldsymbol{y}_k - \boldsymbol{F}\bar{\boldsymbol{x}}_k)^T + \boldsymbol{F}\boldsymbol{\Gamma}^{-1}\boldsymbol{F}^T \tag{54}$$

を得る．

式 (54) をそのまま使うのではなく，計算しやすいように変形する．式 (54) の右辺第 1 項は式 (22) を用いれば

$$\frac{1}{K}\sum_{k=1}^{K}(\boldsymbol{y}_k - \boldsymbol{F}\bar{\boldsymbol{x}}_k)(\boldsymbol{y}_k - \boldsymbol{F}\bar{\boldsymbol{x}}_k)^T = \frac{1}{K}\sum_{k=1}^{K}(\boldsymbol{y}_k - \boldsymbol{F}\boldsymbol{\Omega}\boldsymbol{y}_k)(\boldsymbol{y}_k - \boldsymbol{F}\boldsymbol{\Omega}\boldsymbol{y}_k)^T = \boldsymbol{\Pi}\boldsymbol{R}_{yy}\boldsymbol{\Pi}^T \tag{55}$$

である．ここで

$$\boldsymbol{\Pi} = \boldsymbol{I} - \boldsymbol{F}\boldsymbol{\Omega} = \boldsymbol{I} - \boldsymbol{F}\boldsymbol{\Upsilon}\boldsymbol{F}^T\boldsymbol{\Sigma}_y^{-1} \tag{56}$$

$$\boldsymbol{R}_{yy} = \frac{1}{K}\sum_{k=1}^{K}\boldsymbol{y}_k\boldsymbol{y}_k^T \tag{57}$$

である．

式 (54) の右辺第 2 項については，まず，$\boldsymbol{\Gamma}^{-1}$（式 (18)）を以下のように変形する．$\boldsymbol{\Gamma}^{-1}$ は，逆行列公式より，

$$\boldsymbol{\Gamma}^{-1} = \left(\boldsymbol{\Upsilon}^{-1} + \boldsymbol{F}^T \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1} \boldsymbol{F}\right)^{-1} = \boldsymbol{\Upsilon} - \boldsymbol{\Upsilon}\boldsymbol{F}^T (\boldsymbol{\Lambda} + \boldsymbol{F}\boldsymbol{\Upsilon}\boldsymbol{F}^T)^{-1} \boldsymbol{F}\boldsymbol{\Upsilon} = \boldsymbol{\Upsilon} - \boldsymbol{\Upsilon}\boldsymbol{F}^T \boldsymbol{\Sigma}_y^{-1} \boldsymbol{F}\boldsymbol{\Upsilon} \tag{58}$$

を得る．したがって，式 (54) の右辺第 2 項は

$$\begin{aligned} \boldsymbol{F}\boldsymbol{\Gamma}^{-1}\boldsymbol{F}^T &= \boldsymbol{F}\left(\boldsymbol{\Upsilon} - \boldsymbol{\Upsilon}\boldsymbol{F}^T \boldsymbol{\Sigma}_y^{-1} \boldsymbol{F}\boldsymbol{\Upsilon}\right)\boldsymbol{F}^T = \boldsymbol{F}\boldsymbol{\Upsilon}\boldsymbol{F}^T - \boldsymbol{F}\boldsymbol{\Upsilon}\boldsymbol{F}^T \boldsymbol{\Sigma}_y^{-1} \boldsymbol{F}\boldsymbol{\Upsilon}\boldsymbol{F}^T \\ &= \left(\boldsymbol{I} - \boldsymbol{F}\boldsymbol{\Upsilon}\boldsymbol{F}^T \boldsymbol{\Sigma}_y^{-1}\right)\boldsymbol{F}\boldsymbol{\Upsilon}\boldsymbol{F}^T = (\boldsymbol{I} - \boldsymbol{F}\boldsymbol{\Omega})\,\boldsymbol{F}\boldsymbol{\Upsilon}\boldsymbol{F}^T = \boldsymbol{\Pi}\boldsymbol{F}\boldsymbol{\Upsilon}\boldsymbol{F}^T \end{aligned} \tag{59}$$

となるので，結局，$\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}$ の更新式 (54) は

$$\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{\varepsilon}} = \boldsymbol{\Pi}\boldsymbol{R}_{yy}\boldsymbol{\Pi}^T + \boldsymbol{\Pi}\boldsymbol{F}\boldsymbol{\Upsilon}\boldsymbol{F}^T \tag{60}$$

と表される．通常は $\boldsymbol{\varepsilon}$ を対角ノイズ（diagonal noise）と仮定しているので，$\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}$ を対角行列として，更新式は

$$\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{\varepsilon}} = \mathrm{diag}\left[\boldsymbol{\Pi}\boldsymbol{R}_{yy}\boldsymbol{\Pi}^T + \boldsymbol{\Pi}\boldsymbol{F}\boldsymbol{\Upsilon}\boldsymbol{F}^T\right] \tag{61}$$

となる[6]．

5 SBL ビームフォーミング

SBL ビームフォーミングは SBL アルゴリズムから出力されたモデルデータ共分散行列 $\boldsymbol{\Sigma}_y$ を用いてビームフォーマーを構成する信号源再構成手法である．従来のビームフォーマーは，サンプルデータ共分散行列 $\boldsymbol{R}$ を用いて構成した重み行列：

$$\boldsymbol{W}(\boldsymbol{r}) = \boldsymbol{R}^{-1}\boldsymbol{L}(\boldsymbol{r}) \left[\boldsymbol{L}^T(\boldsymbol{r})\boldsymbol{R}^{-1}\boldsymbol{L}(\boldsymbol{r}_)\right]^{-1} \tag{62}$$

を用いて，位置 $\boldsymbol{r}$ におけるソースベクトルを

$$\widehat{\boldsymbol{s}}(\boldsymbol{r}, t) = \boldsymbol{W}^T(\boldsymbol{r})\boldsymbol{y}(t) = \left[\boldsymbol{L}^T(\boldsymbol{r})\boldsymbol{R}^{-1}\boldsymbol{L}(\boldsymbol{r}_)\right]^{-1} \boldsymbol{L}^T(\boldsymbol{r})\boldsymbol{R}^{-1}\boldsymbol{y}(t) \tag{63}$$

として推定する．SBL ビームフォーミングは，サンプル共分散行列 $\boldsymbol{R}$ の代わりに，SBL アルゴリズムの出力として求まるモデルデータ共分散行列 $\boldsymbol{\Sigma}_y$ を用いて重み行列

$$\boldsymbol{W}(\boldsymbol{r}) = \boldsymbol{\Sigma}_y^{-1}\boldsymbol{L}(\boldsymbol{r}) \left[\boldsymbol{L}^T(\boldsymbol{r})\boldsymbol{\Sigma}_y^{-1}\boldsymbol{L}(\boldsymbol{r}_)\right]^{-1} \tag{64}$$

を求め，位置 $\boldsymbol{r}$ におけるソースベクトルを

$$\widehat{\boldsymbol{s}}(\boldsymbol{r}, t) = \boldsymbol{W}^T(\boldsymbol{r})\boldsymbol{y}(t) = \left[\boldsymbol{L}^T(\boldsymbol{r})\boldsymbol{\Sigma}_y^{-1}\boldsymbol{L}(\boldsymbol{r}_)\right]^{-1} \boldsymbol{L}^T(\boldsymbol{r})\boldsymbol{\Sigma}_y^{-1}\boldsymbol{y}(t) \tag{65}$$

[6] ここで，行列を $\boldsymbol{A}$ として $\boldsymbol{B} = \mathrm{diag}(\boldsymbol{A})$ とは，「$\boldsymbol{A}$ の対角成分を対角成分に持つ対角行列を $\boldsymbol{B}$ とする」の意味である．

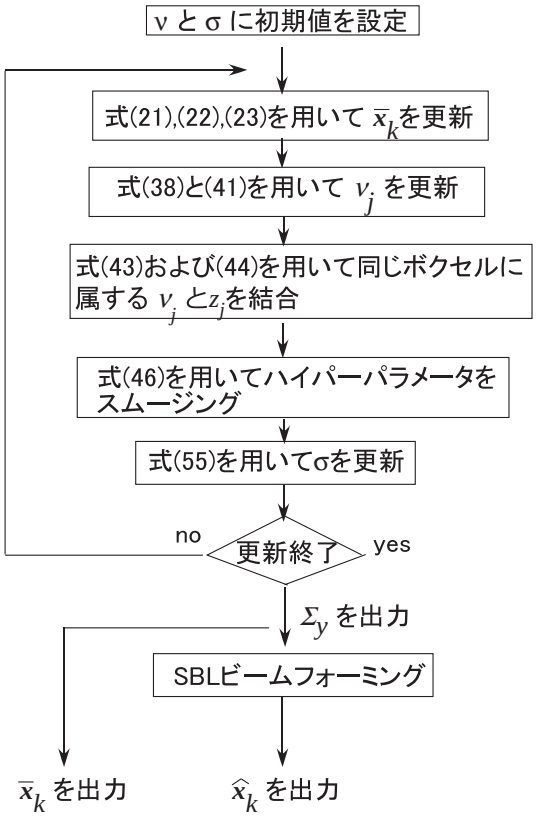


図 1: RENSx アルゴリズムの流れ．ベクトル $\boldsymbol{\sigma}$ はノイズ共分散行列 $\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}$ の対角成分を要素に持つ列ベクトルであり，$\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} = diag(\boldsymbol{\sigma})$ の関係がある．

と推定する．

SBL ビームフォーマーは次の 2 つの点で，従来のビームフォーマーよりも優れている．サンプル共分散行列 $\boldsymbol{R}$ を精度良く求めるためには多数のタイムサンプル（すなわち大きな K の値）を必要とする．そのため，従来のビームフォーマーは多数のタイムサンプルを含むデータ時間窓を必要とする．これがためにビームフォーマーは多数のタイムサンプルを含むデータ時間窓を確保できない計測データ，例えば誘発脳磁界データなどにはあまり用いられることはなかった．一方で，SBL アルゴリズムによるモデルデータ共分散行列の推定ではこのような制約は存在せず，少数のタイムサンプルからもモデルデータ共分散行列の推定が可能である．

さらに，ビームフォーマーには複数の信号源活動に時間的な相関があると信号源推定結果に大きな誤差を生じることが知られている．これは，データサンプル共分散行列が信号源相関の影響を大きく受け，その信号およびノイズ部分空間が縮退してしまうためである．一方，SBL アルゴリズムにより求まるモデルデータ共分散行列は信号源活動の相関には影響を受けなため，信号源活動に大きな相関があっても信号源活動の情報を保ったモデルデータ共分散行列の推定が可能である．

付録 C に SBL ビームフォーマーに関する実験結果を発表したポスター（Biomag2016,Soeul,Korea）を示す．このポスターは特に SBL ビームフォーマーの multi-resolution capability に関するコンピュータシミュレーションの報告であり，low-resolution ボクセル（ボクセル間隔：15mm）でモデルデータ共分散行列を求め，high-resolution ボクセル（ボクセル間隔：2mm）でビームフォーマ再構成を行うことで，low-resolution ボクセルでの再構成よりはるかに高分解能の再構成結果を得ることができることを報告している．

6 RENSx アルゴリズムーまとめ

RENSx アルゴリズムを図 1 にまとめた．RENSx アルゴリズムはまず SBL アルゴリズムを実行する．この部分では，まずハイパーパラメータ $\boldsymbol{\nu}$ と $\boldsymbol{\sigma}$ に初期値を与える．ここで，$\boldsymbol{\sigma}$ はノイズ共分散行列 $\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}$ の対角成分を要素に持つ列ベクトルであり，ノイズ共分散行列は $\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} = diag(\boldsymbol{\sigma})$ として求める．次に，$\boldsymbol{\nu}$ と $\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}$ を用いて事後平均 $\bar{\boldsymbol{x}}_k$（$k = 1, \ldots, K$）を求める．さらに，この $\bar{\boldsymbol{x}}_k$ を用いて，$\boldsymbol{\nu}$ と $\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}$ を更新する．以上の手順をある条件が満たされるまで行う．SBL アルゴリズムの出力は事後平均 $\bar{\boldsymbol{x}}_k$（$k = 1, \ldots, K$）とモデルデータ共分散行列 $\boldsymbol{\Sigma}_y$（式 (21)）である．RENSx アルゴリズムにおいては，SBL アルゴリズムの出力であるモデルデータ共分散行列 $\boldsymbol{\Sigma}_y$ を用いて SBL ビームフォーミングを行う．アルゴリズムの最終的な出力は SBL から出力される SBL ボクセルタイムコース $\bar{\boldsymbol{x}}_k$（$k = 1, \ldots, K$）と，ビームフォーマーの出力 $\bar{\boldsymbol{x}}_k$（$k = 1, \ldots, K$）である．

7 参考文献

以下に SBL アルゴリズムの説明がある．

1. Electromagnetic Brain Imaging, Springer 2015.
2. ベイズ信号処理，共立出版 2015.
3. 信号処理のための線形代数入門，2019.
4. シグナルアナリシス社技術資料「スパースベイズ推定」
   `http://www.signalanalysis.jp/pdfs/SBL_note.pdf`
5. シグナルアナリシス社技術資料「ベイズミニマムノルム推定」
   `http://www.signalanalysis.jp/pdfs/BMN_note.pdf`

A Appendix: マトリックスSBLアルゴリズム

A.1 確率モデルと学習測

本ノートの3節で述べたSBLアルゴリズムは各ボクセルの共分散行列をスカラー倍の単位行列に仮定したおり，スカラータイプのSBLアルゴリズムと呼ばれる．これに対し，ボクセル共分散行列に特別な仮定を置かないSBLアルゴリズムを構築することも可能である．このアルゴリズムは（いわば）マトリックスタイプのSBLアルゴリズムであり，matrix champagneと呼ばれる．

このアルゴリズムでは各ボクセルにおける j 番目のソース $\boldsymbol{s}(\boldsymbol{r}_j, t)$ に対して事前分布を，

$$p(\boldsymbol{s}_j(t)) = \mathcal{N}(\boldsymbol{s}_j(t)|\boldsymbol{0}, \boldsymbol{\Upsilon}_j) \tag{66}$$

とする．簡単のため $\boldsymbol{s}(\boldsymbol{r}_j, t)$ を $\boldsymbol{s}_j(t)$ と書いた．ここで，共分散行列 $\boldsymbol{\Upsilon}_j$ は 3×3 の行列である．各ボクセルにおけるソースベクトル $\boldsymbol{s}_j(t)$ は（固定の）任意の向きを取りえるので対角行列と仮定はできない．したがって，ボクセルベクトル $\boldsymbol{x}_k$ に対する事前分布は

$$p(\boldsymbol{x}_k|\boldsymbol{\Upsilon}) = \prod_{j=1}^{N} \mathcal{N}(\boldsymbol{s}_j(t_k)|0, \boldsymbol{\Upsilon}_j) = \mathcal{N}(\boldsymbol{x}_k|\boldsymbol{0}, \boldsymbol{\Upsilon}) \tag{67}$$

で表される．ボクセルベクトルの事前共分散行列 $\boldsymbol{\Upsilon}$ は $3N \times 3N$ のブロック対角行列で

$$\boldsymbol{\Upsilon} = \begin{bmatrix} \boldsymbol{\Upsilon}_1 & 0 & \cdots & 0 \\ 0 & \boldsymbol{\Upsilon}_2 & \cdots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{\Upsilon}_N \end{bmatrix}. \tag{68}$$

と表される．

この事前分布と式 (12) のノイズの仮定のもとで，事後分布の平均はやはり

$$\bar{\boldsymbol{x}}_k = \boldsymbol{\Upsilon} \boldsymbol{F}^T \boldsymbol{\Sigma}_y^{-1} \boldsymbol{y}_k \tag{69}$$

と求まる．個々のボクセルにおけるソースベクトルの事後平均は

$$\bar{\boldsymbol{s}}_j(t_k) = \boldsymbol{\Upsilon}_j \boldsymbol{L}_j^T \boldsymbol{\Sigma}_y^{-1} \boldsymbol{y}_k \tag{70}$$

である．ここで，モデルデータ共分散行列 $\boldsymbol{\Sigma}_y$ は

$$\boldsymbol{\Sigma}_y = \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \sum_{j=1}^{N} \boldsymbol{L}_j \boldsymbol{\Upsilon}_j \boldsymbol{L}_j^T \tag{71}$$

である．ただし，$\boldsymbol{L}_j = \boldsymbol{L}(\boldsymbol{r}_j)$ と書いた．

式 (33) に対応したコスト関数は

$$\begin{aligned}\mathcal{F}(\boldsymbol{\Upsilon}) &= -(2/K)\log p(\boldsymbol{y}|\boldsymbol{\Upsilon}) \\ &= \log|\boldsymbol{\Sigma}_y| + \frac{1}{K}\sum_{k=1}^{K}\left[(\boldsymbol{y}_k - \boldsymbol{F}\bar{\boldsymbol{x}}_k)^T\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1}(\boldsymbol{y}_k - \boldsymbol{F}\bar{\boldsymbol{x}}_k) + \sum_{j=1}^{N}\bar{\boldsymbol{s}}_j^T(t_k)\boldsymbol{\Upsilon}_j^{-1}\bar{\boldsymbol{s}}_j(t_k)\right] \quad (72)\end{aligned}$$

である．ここで，式 (34) に対応して，

$$\sum_{j=1}^{N}\mathrm{tr}\left(\boldsymbol{Z}_j^T\boldsymbol{\Upsilon}_j\right) - Z_0 \geq \log|\boldsymbol{\Sigma}_y| \quad (73)$$

を満たす 3×3 の行列 $\boldsymbol{Z}_j$（$j=1,\ldots,N$）が必ず存在する．したがって，代替コスト関数を以下のように定義すれば[7]

$$\begin{aligned}\widetilde{\mathcal{F}}(\boldsymbol{\Upsilon},\boldsymbol{Z}) = \frac{1}{K}\sum_{k=1}^{K}&\left[(\boldsymbol{y}_k - \boldsymbol{F}\bar{\boldsymbol{x}}_k)^T\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1}(\boldsymbol{y}_k - \boldsymbol{F}\bar{\boldsymbol{x}}_k) + \sum_{j=1}^{N}\bar{\boldsymbol{s}}_j^T(t_k)\boldsymbol{\Upsilon}_j^{-1}\bar{\boldsymbol{s}}_j(t_k)\right] \\ &+ \sum_{j=1}^{N}\mathrm{tr}\left(\boldsymbol{Z}_j^T\boldsymbol{\Upsilon}_j\right) - Z_0 \quad (74)\end{aligned}$$

このコスト関数は必ず

$$\widetilde{\mathcal{F}}(\boldsymbol{\Upsilon},\boldsymbol{Z}) \geq \mathcal{F}(\boldsymbol{\Upsilon})$$

であり，$\widetilde{\mathcal{F}}(\boldsymbol{\Upsilon},\boldsymbol{Z})$ を $\boldsymbol{\Upsilon}$ と $\boldsymbol{Z}$ で最小化すれば，その $\boldsymbol{\Upsilon}$ は必ず $\mathcal{F}(\boldsymbol{\Upsilon})$ も最小化する．

ハイパーパラメータ $\boldsymbol{Z}_j$ と vUp_j の更新式は以下のように導出される．（これらの導出は Electromagnetic Brain Imaging の 4.8 節にある．）ハイパーパラメータ $\boldsymbol{Z}_j$ の更新式は

$$\widehat{\boldsymbol{Z}}_j = \frac{\partial}{\partial\boldsymbol{\Upsilon}_j}\log|\boldsymbol{\Sigma}_y| \quad (75)$$

から求まり，右辺の計算を行うと

$$\widehat{\boldsymbol{Z}}_j = \boldsymbol{L}_j^T\boldsymbol{\Sigma}_y^{-1}\boldsymbol{L}_j \quad (76)$$

である．また，$\boldsymbol{\Upsilon}_j$ の更新式は

$$\widehat{\boldsymbol{\Upsilon}}_j = \underset{\boldsymbol{\Upsilon}_j}{\mathrm{argmin}}\left[\frac{1}{K}\sum_{k=1}^{K}\bar{\boldsymbol{s}}_j^T(t_k)\boldsymbol{\Upsilon}_j^{-1}\bar{\boldsymbol{s}}_j(t_k) + \mathrm{tr}\left(\boldsymbol{Z}_j^T\boldsymbol{\Upsilon}_j\right)\right] \quad (77)$$

から求まる．右辺の最小値を与える $\boldsymbol{Z}_j$ は

$$\widehat{\boldsymbol{\Upsilon}}_j = \boldsymbol{Z}_j^{-1/2}\left[\boldsymbol{Z}_j^{1/2}\left[\frac{1}{K}\sum_{k=1}^{K}\bar{\boldsymbol{s}}_j(t_k)\bar{\boldsymbol{s}}_j^T(t_k)\right]\boldsymbol{Z}_j^{1/2}\right]^{1/2}\boldsymbol{Z}_j^{-1/2} \quad (78)$$

[7] $\boldsymbol{Z}_1,\ldots,\boldsymbol{Z}_N$ をまとめて $\boldsymbol{Z}$ す.

と求まる．

ここで述べた matrix champagne はボクセル共分散構造に特別な仮定を置かない，言ってみれば理論的には正統な SBL アルゴリズムであるが，第 3.3.2 節で述べたスカラータイプの SBL アルゴリズムに対して，明らかにアルゴリズムが複雑（スカラー計算が行列計算になる等）になり，その分計算時間を必要とする[8]．その一方で,（理論的な正当性はともかく）実用的にはスカラータイプの SBL アルゴリズムに対して特段の利点は見当たらない．

A.2 Hyperparameter Tying のマトリックス SBL からの導出

マトリックスタイプの SBL はボクセルの共分散構造に何の仮定も置かないで導かれるアルゴリズムである．それでは，ボクセルの共分散行列をスカラーと仮定し，第 3.4 節で導入した Hyperparameter Tying と等価な式が更新式として導かれることを示す．

まず，ボクセルベクトルの事前共分散行列 $\boldsymbol{\Upsilon}$ は $3N \times 3N$ のブロック対角行列で

$$\boldsymbol{\Upsilon} = \begin{bmatrix} \mu_1 \boldsymbol{I}_3 & 0 & \cdots & 0 \\ 0 & \mu_2 \boldsymbol{I}_3 & \cdots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & \mu_N \boldsymbol{I}_3 \end{bmatrix}. \tag{79}$$

と仮定する．ここで，$\boldsymbol{I}_3$ は 3×3 の単位行列を表す．各ボクセルのソースベクトル $\boldsymbol{s}_j(t)$ の事後平均は，したがって,

$$\bar{\boldsymbol{s}}_j(t_k) = \mu_j \boldsymbol{L}_j^T \boldsymbol{\Sigma}_y^{-1} \boldsymbol{y}_k \tag{80}$$

となる．モデルデータ共分散行列は，この場合,

$$\boldsymbol{\Sigma}_y = \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \sum_{j=1}^{N} \mu_j \boldsymbol{L}_j \boldsymbol{L}_j^T \tag{81}$$

である．

したがって，コスト関数は

$$\mathcal{F}(\mu_1, \ldots, \mu_N) = \\ = \log |\boldsymbol{\Sigma}_y| + \frac{1}{K} \sum_{k=1}^{K} \left[(\boldsymbol{y}_k - \boldsymbol{F}\bar{\boldsymbol{x}}_k)^T \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1} (\boldsymbol{y}_k - \boldsymbol{F}\bar{\boldsymbol{x}}_k) + \sum_{j=1}^{N} \mu_j^{-1} \bar{\boldsymbol{s}}_j^T(t_k) \bar{\boldsymbol{s}}_j(t_k) \right] \tag{82}$$

となる．式 (73) に $\boldsymbol{\Upsilon}_j = \mu_j \boldsymbol{I}_3$ と $\boldsymbol{Z}_j = z_j \boldsymbol{I}_3$ を代入すれば,

$$\sum_{j=1}^{N} z_j \mu_j - Z_0 = \boldsymbol{z}^T \boldsymbol{\mu} - Z_0 \geq \log |\boldsymbol{\Sigma}_y| \tag{83}$$

[8]以前の検討では計算時間はスカラーアルゴリズムに比べ 10 倍程度必要であった．

の関係を満たす．ここで，$\boldsymbol{z} = [z_1, \ldots, z_N]$ および $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_N]$ とした．したがって，代替コスト関数を

$$\widetilde{\mathcal{F}}(\boldsymbol{\mu}, \boldsymbol{z}) = \boldsymbol{z}^T \boldsymbol{\mu} - Z_0 + \frac{1}{K} \sum_{k=1}^{K} \left[(\boldsymbol{y}_k - \boldsymbol{F}\bar{\boldsymbol{x}}_k)^T \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1} (\boldsymbol{y}_k - \boldsymbol{F}\bar{\boldsymbol{x}}_k) + \sum_{j=1}^{N} \mu_j^{-1} \bar{\boldsymbol{s}}_j^T(t_k) \bar{\boldsymbol{s}}_j(t_k) \right] \tag{84}$$

とすれば，やはり，

$$\widetilde{\mathcal{F}}(\boldsymbol{\mu}, \boldsymbol{z}) \geq \mathcal{F}(\boldsymbol{\nu}) \tag{85}$$

が成り立つので，$\widetilde{\mathcal{F}}(\boldsymbol{\mu}, \boldsymbol{z})$ を最小とする $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_N]$ は真のコスト関数 $\mathcal{F}(\mu_1, \ldots, \mu_N)$ を最小とする．

実際，$\widetilde{\mathcal{F}}(\boldsymbol{\mu}, \boldsymbol{z})$ を最小とする $\boldsymbol{\mu}$ は

$$\widehat{\boldsymbol{\mu}} = \underset{\boldsymbol{\mu}}{\operatorname{argmin}}\, \widetilde{\mathcal{F}}(\boldsymbol{\mu}, \boldsymbol{z}) = \underset{\boldsymbol{\mu}}{\operatorname{argmin}} \left[\boldsymbol{z}^T \boldsymbol{\mu} + \frac{1}{K} \sum_{k=1}^{K} \sum_{j=1}^{N} \mu_j^{-1} \bar{\boldsymbol{s}}_j^T(t_k) \bar{\boldsymbol{s}}_j(t_k) \right]$$

として求める．

$$\frac{\partial}{\partial \mu_j} \widetilde{\mathcal{F}}(\boldsymbol{\mu}, \boldsymbol{z}) = z_j - \mu_j^{-2} \frac{1}{K} \sum_{k=1}^{K} \bar{\boldsymbol{s}}_j^T(t_k) \bar{\boldsymbol{s}}_j(t_k) = 0$$

より，$\bar{\boldsymbol{s}}_j(t_k) = [\bar{s}_j^x(t_k), \bar{s}_j^y(t_k), \bar{s}_j^z(t_k)]$ とすれば，結局，

$$\widehat{\mu}_j = \sqrt{\frac{\frac{1}{K} \sum_{k=1}^{K} \bar{\boldsymbol{s}}_j^T(t_k) \bar{\boldsymbol{s}}_j(t_k)}{z_j}} = \sqrt{\frac{\frac{1}{K} \sum_{k=1}^{K} \left[\bar{s}_j^x(t_k)^2 + \bar{s}_j^y(t_k)^2 + \bar{s}_j^z(t_k)^2 \right]}{z_j}} \tag{86}$$

を得る．また，$\widetilde{\mathcal{F}}(\boldsymbol{\mu}, \boldsymbol{z})$ を最小とする $\boldsymbol{z}$ は式 (39) から求まる．式 (81) から

$$\frac{\partial}{\partial \mu_j} \boldsymbol{\Sigma}_y = \boldsymbol{L}_j \boldsymbol{L}_j^T$$

であるので，式 (40) を用いて，

$$\widehat{z}_j = \operatorname{tr}\left[\boldsymbol{\Sigma}_y^{-1} \boldsymbol{L}_j \boldsymbol{L}_j^T \right] = \operatorname{tr}\left[\boldsymbol{L}_j^T \boldsymbol{\Sigma}_y^{-1} \boldsymbol{L}_j \right] = (\boldsymbol{l}_j^x)^T \boldsymbol{\Sigma}_y^{-1} \boldsymbol{l}_j^x + (\boldsymbol{l}_j^y)^T \boldsymbol{\Sigma}_y^{-1} \boldsymbol{l}_j^y + (\boldsymbol{l}_j^z)^T \boldsymbol{\Sigma}_y^{-1} \boldsymbol{l}_j^z \tag{87}$$

を得る．ここで，$\boldsymbol{L}_j = \left[\boldsymbol{l}_j^x, \boldsymbol{l}_j^y, \boldsymbol{l}_j^z \right]$ とした．式 (86) に (87) を代入して，結局

$$\widehat{\nu}_j = \sqrt{\frac{\frac{1}{K} \sum_{k=1}^{K} \left[\bar{s}_j^x(t_k)^2 + \bar{s}_j^y(t_k)^2 + \bar{s}_j^z(t_k)^2 \right]}{(\boldsymbol{l}_j^x)^T \boldsymbol{\Sigma}_y^{-1} \boldsymbol{l}_j^x + (\boldsymbol{l}_j^y)^T \boldsymbol{\Sigma}_y^{-1} \boldsymbol{l}_j^y + (\boldsymbol{l}_j^z)^T \boldsymbol{\Sigma}_y^{-1} \boldsymbol{l}_j^z}} \tag{88}$$

と求まる．式 (88) がハイパーパラメータ ν_j の更新式である．

本節で述べた式 (86) は式 (43) と，式 (88) は式 (44) と同じものであり，結局，3.4 節で導入した Hyperparameter tying を，$\boldsymbol{\Upsilon}$ を式 (79) と仮定することにより，matrix タイプの SBL から導いたことになる．

Hyperparameter tyingに関する発表
Biomag2014におけるポスター

付録B

Hyper-parameter tying: a novel method of controlling the sparsity in sparse Bayes source imaging (Champagne) algorithm

Sekihara K. [1] *Nagarajan S. S.* [2]

[1] *Department of Systems Design and Engineering, Tokyo Metropolitan University, Hino, Tokyo 191-0065, Japan.*

[2] *Biomagnetic Imaging Laboratory, Department of Radiology, University of California, San Francisco, 513 Parnassus Avenue, S362, San Francisco, CA 94143, USA*

Summary: Part I

The sparse Bayes source imaging (Champagne) algorithm has proven to be effective for neuromagnetic imaging [1]. The method produces sparse solutions because a sparsity constraint is integrated in its cost function. However, there are some occasions where the sparsity constraint rather leads to wrong solutions. One such occasion is the estimation of voxel source orientations.

In Champagne algorithm, when the voxel source orientation is known, a diagonal covariance matrix can be used for the Gaussian prior distribution, and this leads to a simple and efficient algorithm. When the voxel source orientation is unknown, a block-diagonal covariance matrix must be used, which prevents from the sparsity constraint being imposed on each component of a source vector. However, the use of such a non-diagonal covariance matrix leads to an algorithm that is significantly complex and computationally expensive.

Summary: Part II

In this paper, we propose a simple method to overcome this difficulty, called the hyper-parameter tying. The proposed method can use a diagonal covariance matrix even when the source orientation is unknown. However, in each step of updating iteration, the source variance parameters of vector components are tied together to correctly estimate the source orientation.

Since a diagonal covariance matrix is used, the algorithm keeps the simplicity and computational efficiency. We discuss why the hyper-parameter tying can control the sparsity by presenting the cost function analysis of the Champagne algorithm[2].

References

Reference

[1] Wipf DP, Owen JP, Attias HT, Sekihara K, Nagarajan SS., Neuroimage. 2010; 49(1) :641-55.

[2] Wipf DP, Nagarajan SS, Advances in Neural Information Processing Systems, 2008

Sparse Bayes source imaging (Champagne) algorithm

Data model: $\boldsymbol{y} = \boldsymbol{F}\boldsymbol{x} + \boldsymbol{\varepsilon}$

Probability model: $p(\boldsymbol{y} \mid \boldsymbol{x}) = N(\boldsymbol{y} \mid \boldsymbol{F}\boldsymbol{x}, \sigma^2 \boldsymbol{I})$

$p(\boldsymbol{x}) = N(\boldsymbol{x} \mid 0, \boldsymbol{\Upsilon})$

Covariance matrix of the prior distribution

$$\boldsymbol{\Upsilon} = \mathrm{diag}[\nu, \nu, \ldots, \nu] \Rightarrow \hat{\boldsymbol{x}} = \arg\min F(\boldsymbol{x}) : F(\boldsymbol{x}) = \frac{1}{\sigma^2}\left\|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}\right\|^2 + \frac{1}{\nu}\left\|\boldsymbol{x}\right\|^2$$

L2 regularized minimum-norm method

$$\boldsymbol{\Upsilon} = \mathrm{diag}[\nu_1, \nu_2, \ldots, \nu_N] = \mathrm{diag}(\boldsymbol{\nu}) \Rightarrow$$

$$\boldsymbol{\nu} = [\nu_1, \nu_2, \ldots, \nu_N]^T$$

$$\hat{\boldsymbol{\nu}} = \arg\min F(\boldsymbol{\nu}) : F(\boldsymbol{\nu}) = \log|\boldsymbol{\Sigma}| + \boldsymbol{y}^T \boldsymbol{\Sigma} \boldsymbol{y}$$

$$\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I} + \boldsymbol{F}\boldsymbol{\Upsilon}\boldsymbol{F}^T$$

$$\hat{\boldsymbol{x}} = \boldsymbol{\Upsilon}\boldsymbol{F}^T \hat{\boldsymbol{\Sigma}}^{-1} \boldsymbol{y}$$

Champagne algorithm

When voxel orientation is unknown:

$$\boldsymbol{F} = [\ldots, \underbrace{\boldsymbol{l}_x(\boldsymbol{r}_j), \boldsymbol{l}_y(\boldsymbol{r}_j), \boldsymbol{l}_z(\boldsymbol{r}_j)}_{\text{lead field for the } j\text{th voxel}}, \ldots]$$

$$\boldsymbol{\Upsilon} = \mathrm{diag}[\nu_1, \nu_2, \ldots, \nu_N] \quad \Rightarrow$$

Leading to the shrinkage over the source vector components.

$$\boldsymbol{\Upsilon} = \mathrm{diag}[\boldsymbol{\Upsilon}_1, \boldsymbol{\Upsilon}_2, \ldots, \boldsymbol{\Upsilon}_N] \quad \Rightarrow$$

Leading to a computer-intensive, slow algorithm, called the matrix champagne.

$\boldsymbol{\Upsilon}_j : 3 \times 3$ submatrix

Hyperparameter tying:

$$\boldsymbol{\Upsilon} = \mathrm{diag}[\ldots, \underbrace{\nu_{3j+1}, \nu_{3j+2}, \nu_{3j+3}}_{\text{hyperparameters for the } j\text{th voxel}}, \ldots]$$

Set the same update value such as $\hat{\nu}_j = (1/3)\sum_{m=1}^{3} \nu_{3j+m}$ to $\nu_{3j+1}, \nu_{3j+2}, \nu_{3j+3}$

Cost function for sparse Bayes source imaging (Champagne algorithm)

$$\hat{\boldsymbol{x}} = \arg\min F(\boldsymbol{x}) : F(\boldsymbol{x}) = \frac{1}{\sigma^2}\left\|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}\right\|^2 + \phi(\boldsymbol{x})$$

$$\phi(\boldsymbol{x}) = \min_{\boldsymbol{\nu}}(\boldsymbol{x}^T \boldsymbol{\Upsilon}^{-1}\boldsymbol{x} + \log|\boldsymbol{\Sigma}|)$$

If we assume that the columns of $\boldsymbol{H}$ are orthogonal,

$$\log|\boldsymbol{\Sigma}| = \sum_{j=1}^{N}\log(\sigma^2 + \nu_j)$$

2-dimensional case

$$\phi(x_1, x_2) = \min_{\nu_1,\nu_2}\sum_{j=1}^{2}\left(\frac{x_j^2}{\nu_j} + \log(\sigma^2 + \nu_j)\right)$$

$$= \sum_{j=1}^{2}\left[\frac{2|x_j|}{\sqrt{x_j^2 + 4\sigma^2} + |x_j|} + \log\left(\sigma^2 + \frac{x_j^2}{2} + \frac{|x_j|}{2}\sqrt{x_j^2 + 4\sigma^2}\right)\right]$$

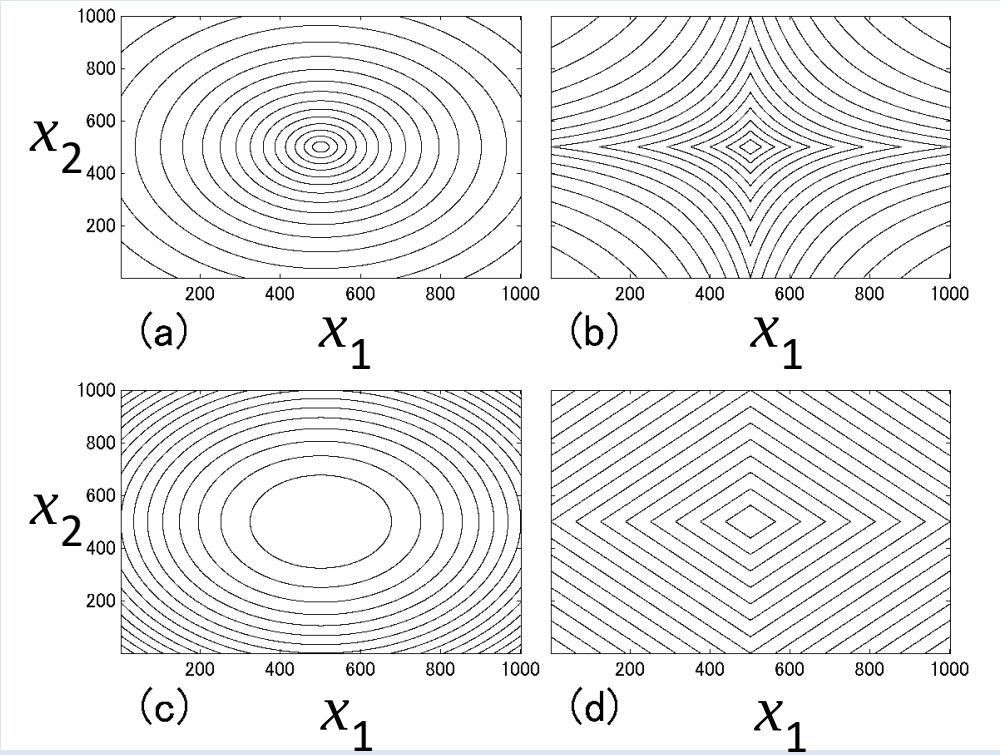
No hyperparameter tying

$$\phi(x_1, x_2) = \min_{\nu}\left(\frac{x_1^2 + x_2^2}{\nu} + 2\log(\sigma^2 + \nu)\right)$$

$$= \left[\frac{4a}{\sqrt{a^2 + 8a\sigma^2} + a} + 2\log\left(\sigma^2 + \frac{a + \sqrt{a^2 + 8a\sigma^2}}{4}\right)\right] : \ (a = x_1^2 + x_2^2)$$

Hyperparameter tying

Plots of cost functions

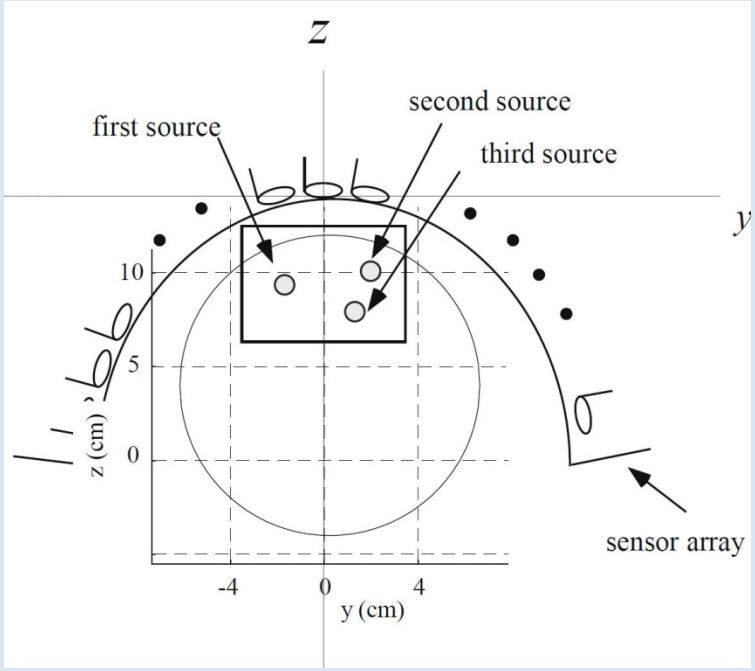


(a) Champagne cost function with tying hyperparameters.
(b) Champagne cost function with no hyperparameter tying.
(c) Cost function for L2-norm regularization.
(d) Cost function for L1-norm regularization.

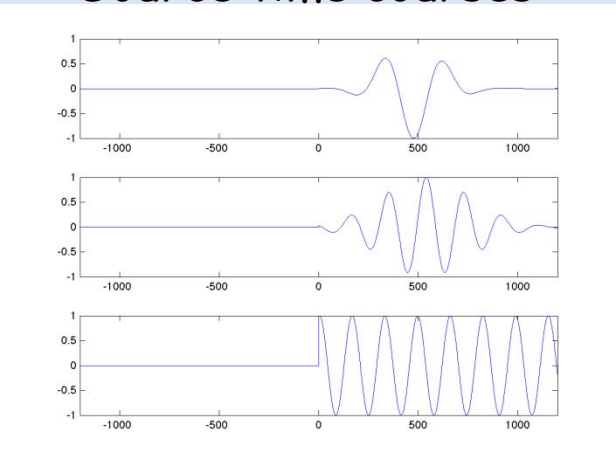
$$\phi(x_1, x_2) = x_1^2 + x_2^2 \text{ : L2-norm regularization}$$

$$\phi(x_1, x_2) = |x_1| + |x_2| \text{: L1-norm regularization}$$

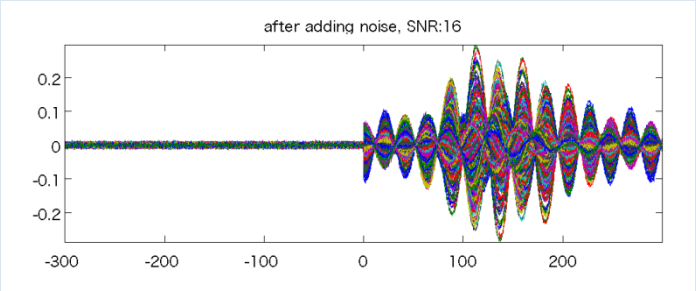
Computer Simulation



Source time courses



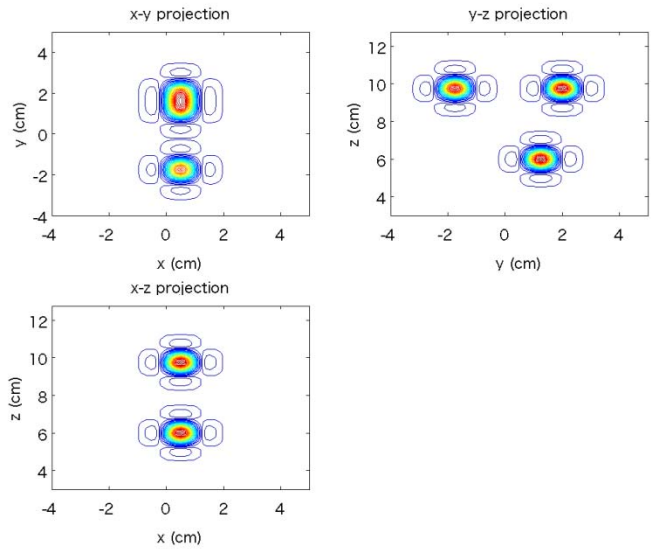
Signal magnetic field



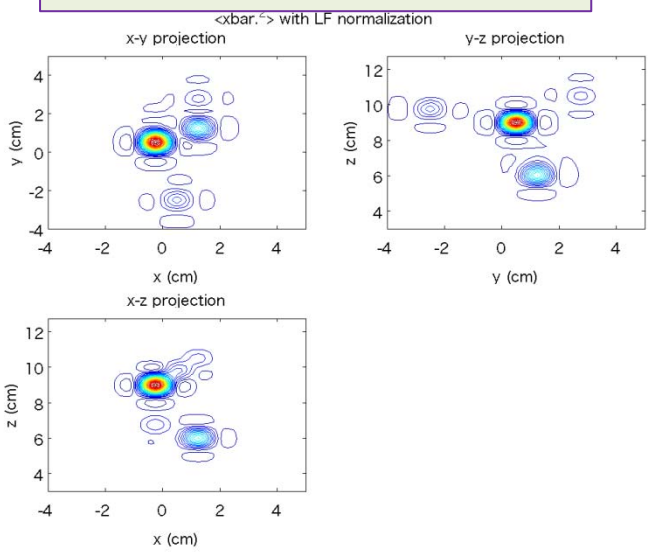
- 275 CTF sensor array
- signal to sensor noise ratio:16

Results of Computer Simulation

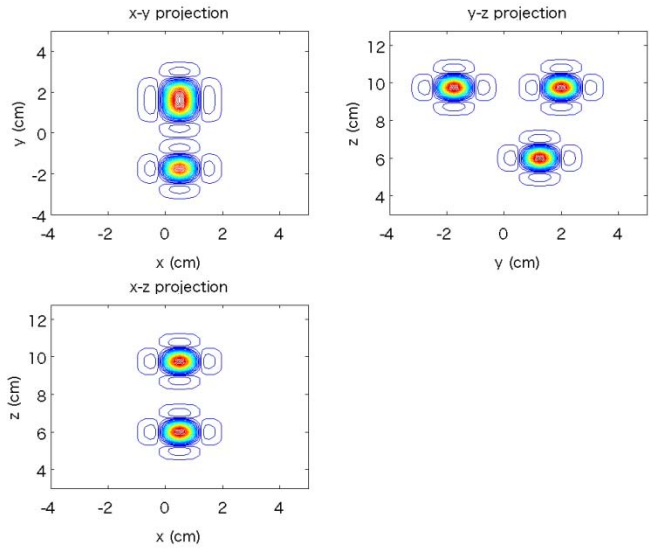
Block-diagonal covariance used



Diagonal covariance used



Diagonal covariance with hyperparameter tying



Champagne with a block-diagonal covariance needs twelve times longer computation time, compared to the champagne algorithm with a diagonal covariance matrix.

SBL beamformerに関する発表
Biomag2016におけるポスター

付録C

Multi-resolution champagne beamformer

Sekihara K. [1,2] *Nagarajan S. S.* [3]

[1] *Tokyo Medical and Dental University, Tokyo, Japan*
[2] *Signal Analysis Inc., Tokyo, Japan*

[3] *University of California, San Francisco, USA*

Summary

The sparse Bayesian source imaging (Champagne) algorithm has proven to be effective for neuromagnetic imaging [1,2]. It is robust to various types of noise and interference, and it can reconstruct multiple sources, even when they are highly correlated [2]. However, one serious problem of the algorithm is that it is computationally expensive because of its iterative nature.

In this paper, we propose a novel algorithm, called multi-resolution champagne beamformer. In the proposed algorithm, the first step estimates the model data covariance by using the sparse Bayesian scheme, and the second step implements the adaptive beamformer source imaging using the data covariance estimated in the first step[3]. The key point of the algorithm is to exploit our empirical findings that the quality of the estimated data covariance does not very much depend on the voxel size in the first step, while the voxel size primarily determines the quality of the source reconstruction in the second step.

Therefore, we can significantly shorten the computational time by using a low-resolution voxel grid in the first step but still obtain the high-quality source images by using a high-resolution grid in the second step.

Computer simulation verifies the performance of the algorithm. Unlike the conventional adaptive beamformer, which uses a sample covariance matrix, the proposed champagne beamformer can be applied to a case where correlated sources exist and a case where only a short time window with a small number of time sample is available. The algorithm may be useful in non-brain electrophysiological imaging such as cardiac imaging in which problems of correlated sources and/or a short time window are common.

References

[1] Wipf DP et al. , Neuroimage. 2010; 49(1): 641-55.

[2] Sekihara S and Nagarajan SS, Electromagnetic brain imaging: a Bayesian perspective, Springer, 2015.

[3] Wipf DP and Nagarajan SS, *Proceedings of the 24th international conference on Machine learning* (pp. 1023-1030). ACM.

Sparse Bayes source imaging (Champagne) algorithm

Data model: $\boldsymbol{y} = \boldsymbol{F}\boldsymbol{x} + \boldsymbol{\varepsilon}$

Probability model: $p(\boldsymbol{y} \mid \boldsymbol{x}) = N(\boldsymbol{y} \mid \boldsymbol{F}\boldsymbol{x}, \sigma^2 \boldsymbol{I})$

$$p(\boldsymbol{x}) = N(\boldsymbol{x} \mid 0, \boldsymbol{\Upsilon})$$

The hyper parameter $\boldsymbol{\Upsilon}$ is obtained through maximizing the marginal likelihood

$$\log p(\boldsymbol{y} \mid \boldsymbol{\Upsilon}) = -\frac{1}{2}\left[\mid \boldsymbol{\Sigma}_y \mid + \boldsymbol{y}^T \boldsymbol{\Sigma}_y^{-1} \boldsymbol{y} \right],$$

where the model data covariance is defined as

$$\boldsymbol{\Sigma}_y = \sigma^2 \boldsymbol{I} + \boldsymbol{F}\boldsymbol{\Upsilon}\boldsymbol{F}^T$$

This maximization can be implemented in several ways, and the one we have developed is described in [1,2].

Champagne beamformer

The champagne beamformer uses the model data covariance derived from the champagne algorithm[3]:

$$\boldsymbol{w}(\boldsymbol{r}) = \frac{\boldsymbol{\Sigma}_y^{-1}\boldsymbol{l}(\boldsymbol{r})}{\boldsymbol{l}^T(\boldsymbol{r})\boldsymbol{\Sigma}_y^{-1}\boldsymbol{l}(\boldsymbol{r})}$$

The conventional adaptive beamformer uses the sample data covariance $\boldsymbol{R}$ derived from a data time window containing many time points[4]:

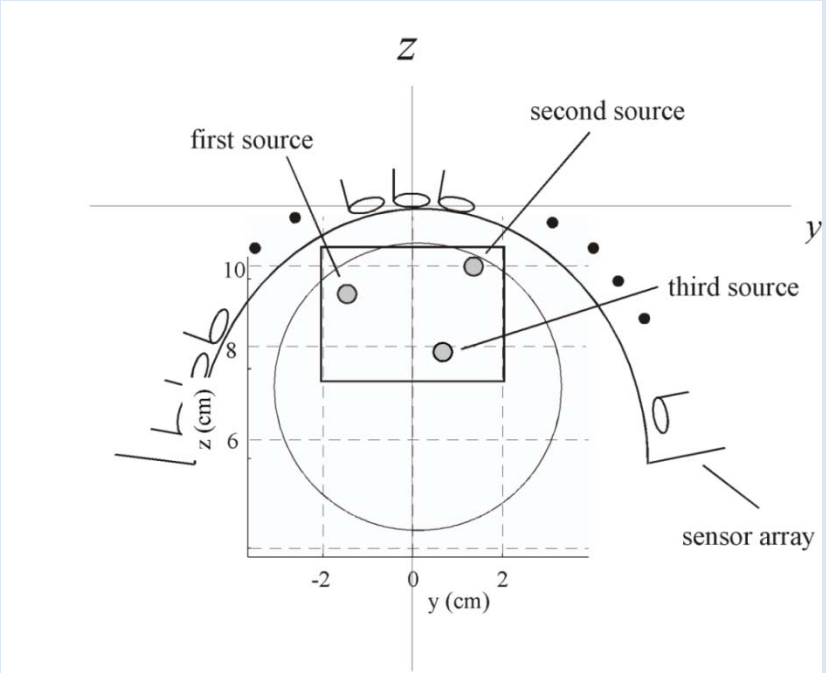
$$\boldsymbol{w}(\boldsymbol{r}) = \frac{\boldsymbol{R}^{-1}\boldsymbol{l}(\boldsymbol{r})}{\boldsymbol{l}^T(\boldsymbol{r})\boldsymbol{R}^{-1}\boldsymbol{l}(\boldsymbol{r})}$$

---

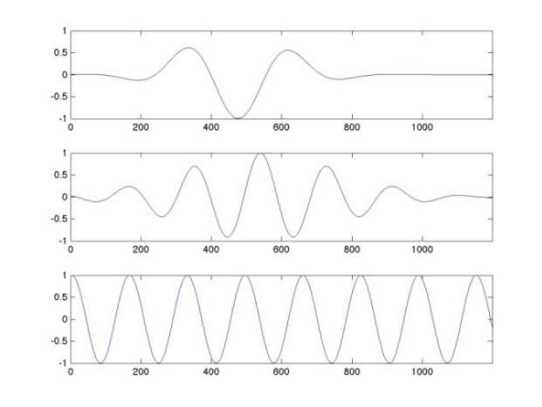
[4] Sekihara K and Nagarajan SS, *Adaptive spatial filters for electromagnetic brain imaging*. Springer, 2008.

Data generation for computer simulation

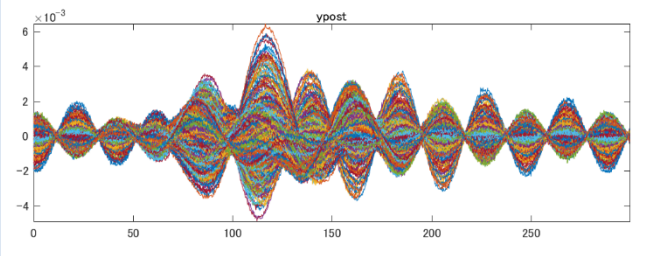
Computer simulation geometry



Time courses assigned to the three sources



Simulated sensor recordings



275 (CTF) whole head sensor array was assumed for data generation.

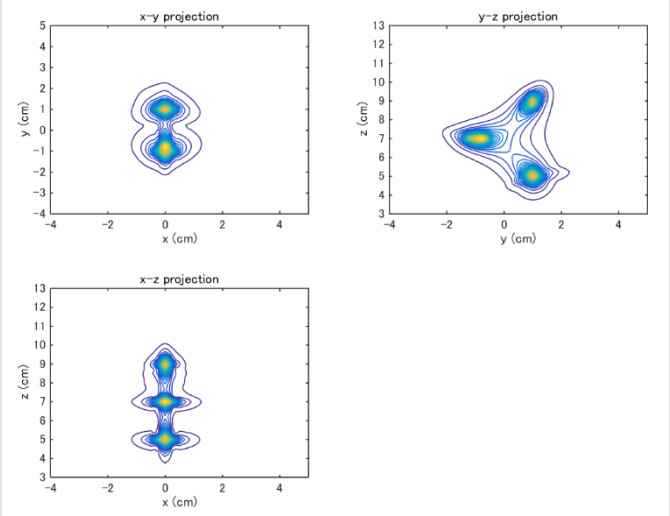
The model data covariance was estimated with voxel grid of 1.5cm.
The beamformer scan was performed with voxel grid of 0.2cm.

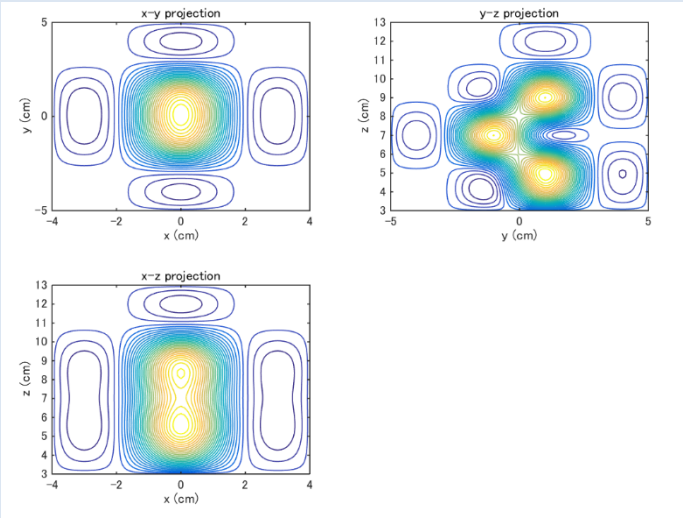
Results of Computer Simulation with no array mismatch

Champagne beamformer images with 2mm voxels

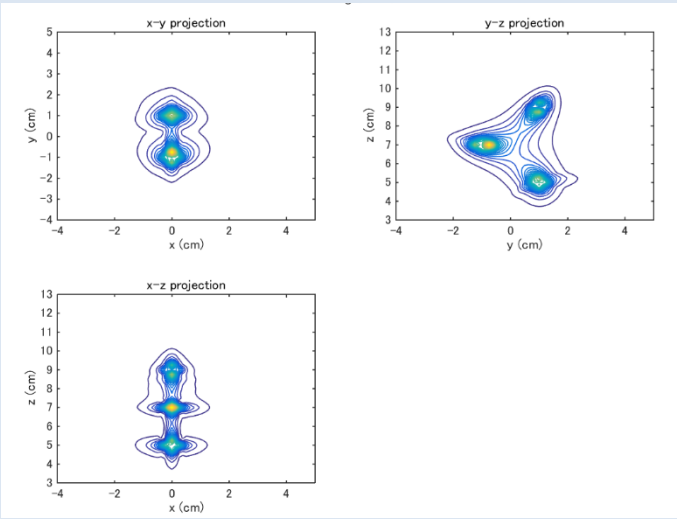
Champagne images with 1.5cm voxels

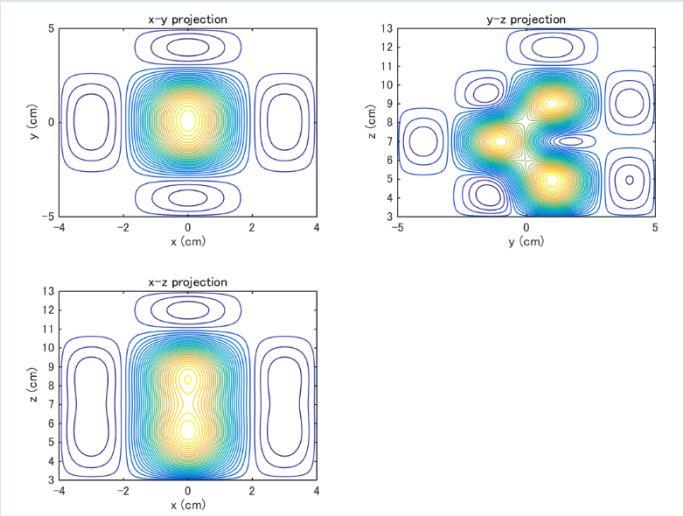
SNR:8





SNR:4





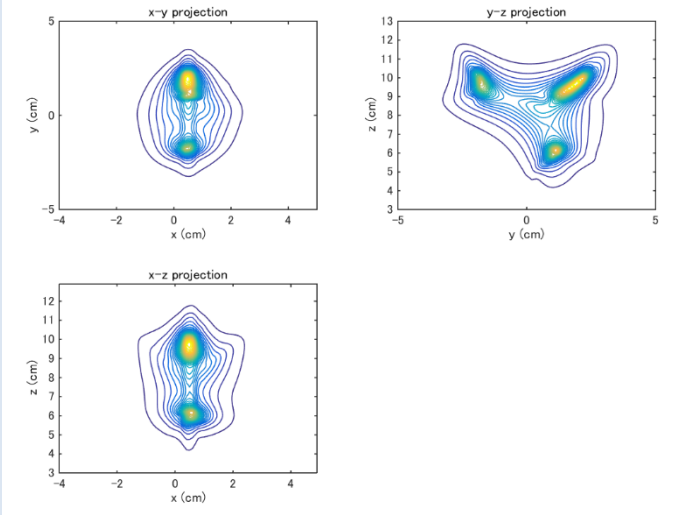
coordinate of three sources: [0 -1 7], [0. 1 9], [0. 1 5] (cm)

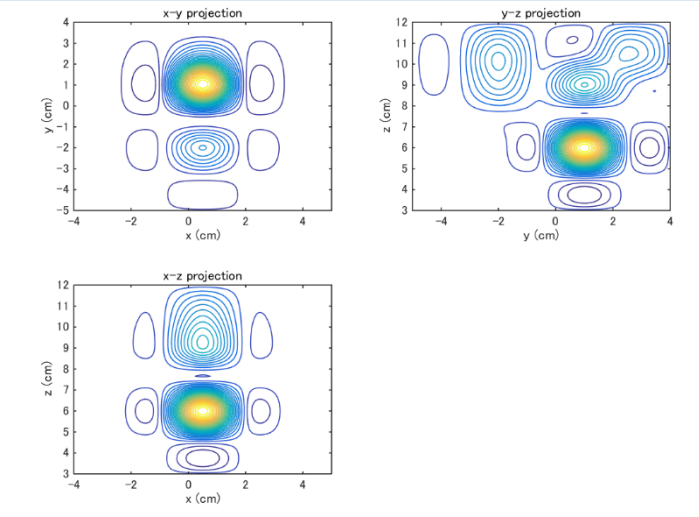
Results of Computer Simulation with array mismatch

Champagne beamformer images with 2mm voxels

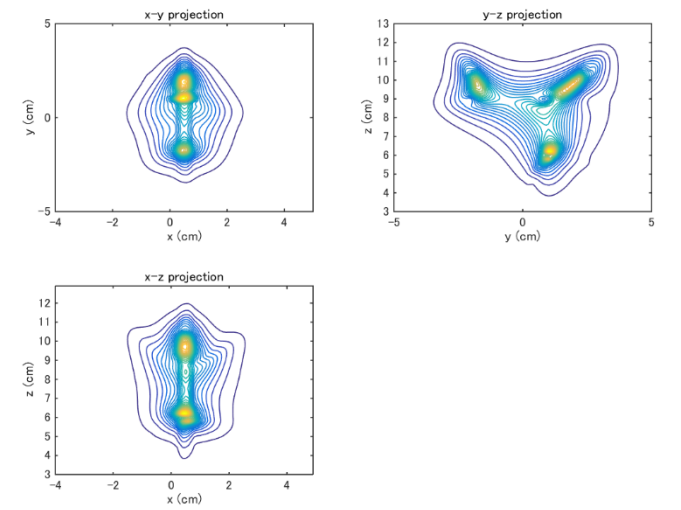
Champagne images with 1.5cm voxels

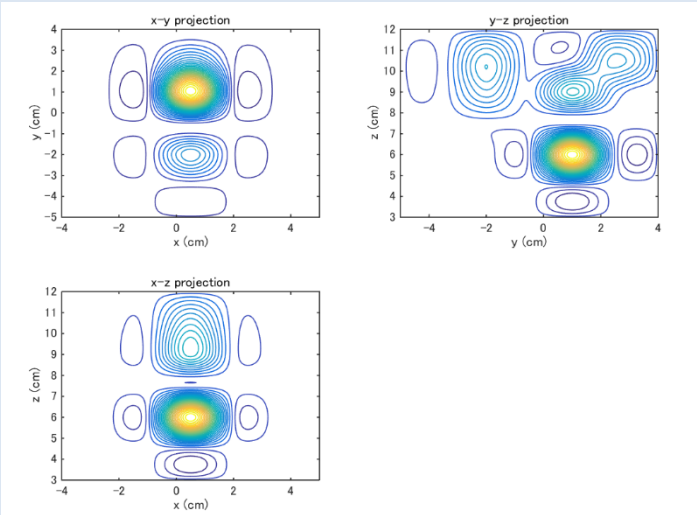
SNR:8





SNR:4





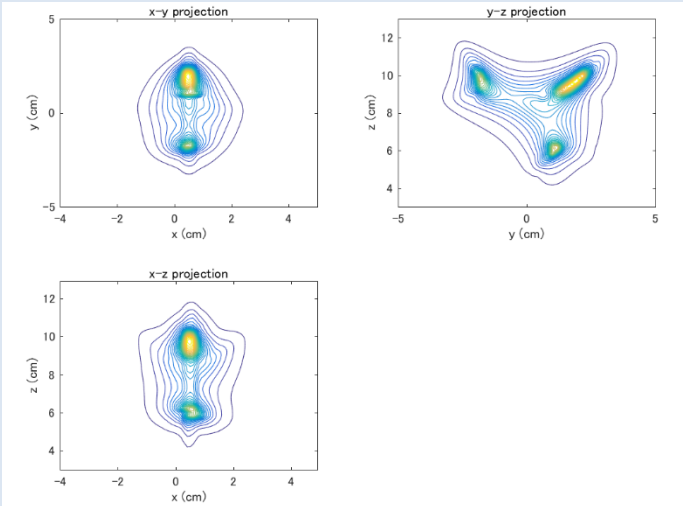
coordinate of three sources: [0.5 -1.75 9.75], [0.5 2 9.75], [0.5 1.25 6] (cm)

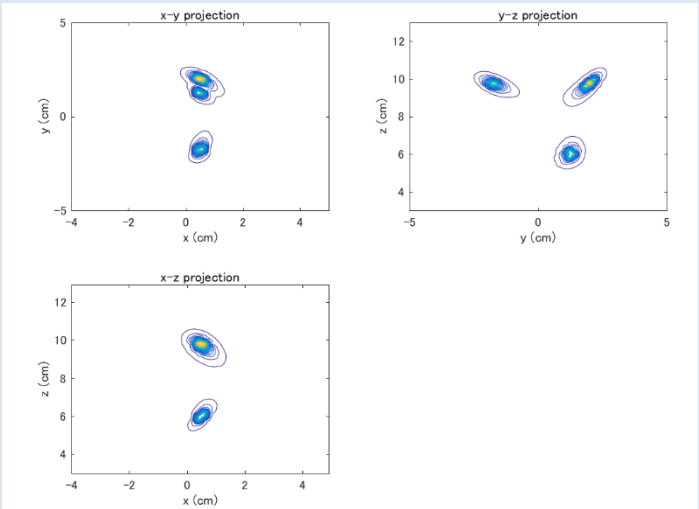
Reconstruction of perfectly correlated sources

Champagne beamformer images

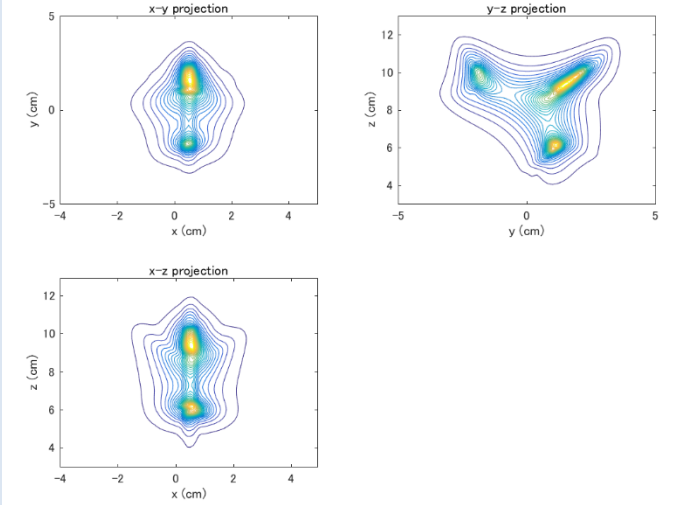
Conventional beamformer images

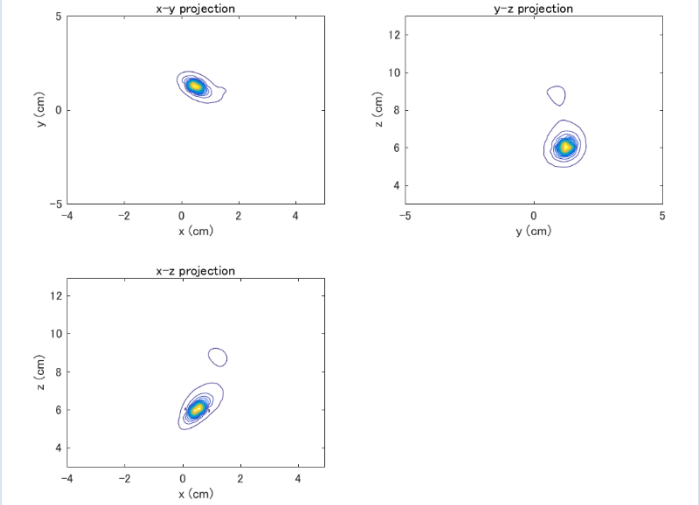
no source correlation





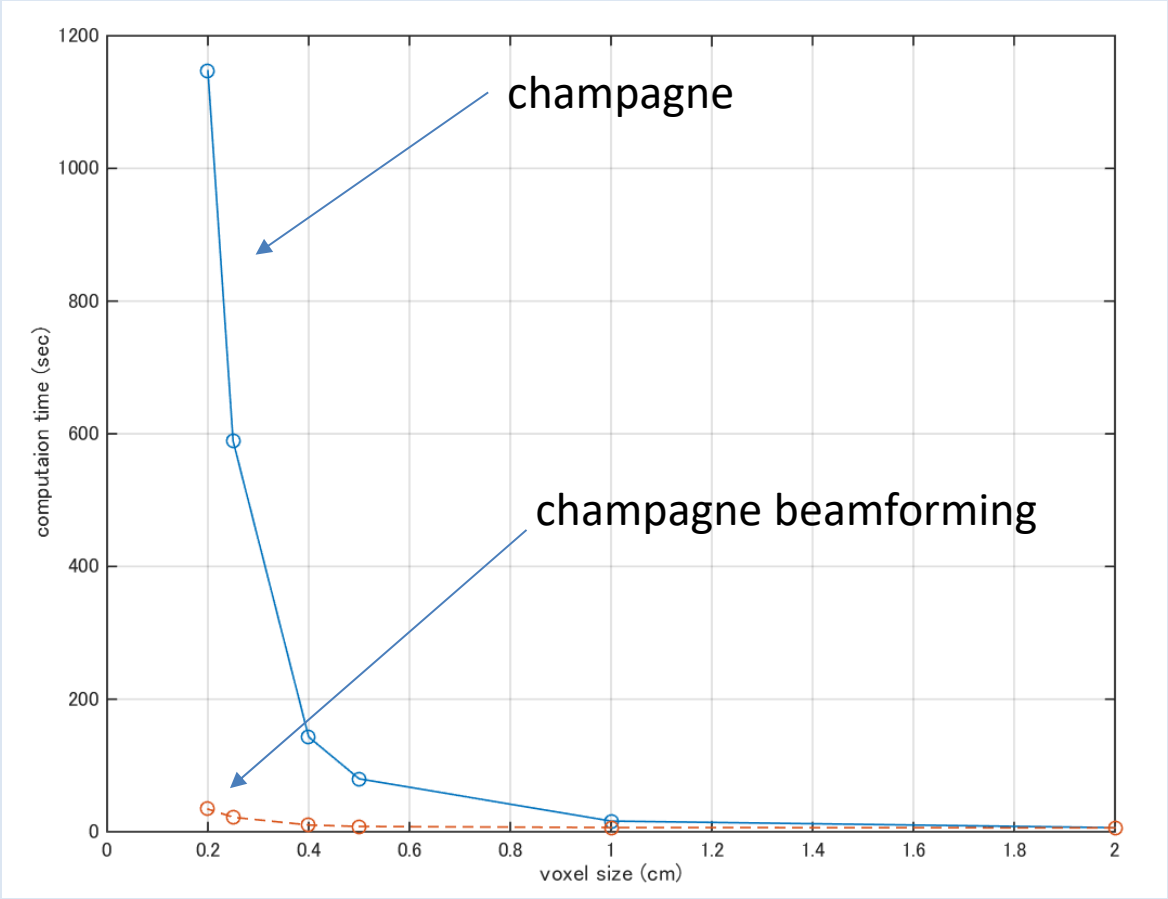
nearly perfect source correlation





The first and the second (the upper two) sources are nearly perfectly correlated in the bottom case.

Comparison of computational time



Condition

# of iteration: 400
# of time points: 1200
# of sensor channels: 275
reconstruction volume: 9cmX9cmX9 cm

Intel® Core i7 3.4GHz
32GB memory
64 bit windows 7
matlab 2014b