Katholieke Universiteit Leuven

Departement Elektrotechniek ESAT-SISTA/TR 2005-120

$\mathbf{AQSES_{VP}}$ – Description of a Variable Projection Implementation for Nonlinear Least Squares with Linear Bounds Constraints, Applied to Accurate Quantification of Short-echo Time Magnetic Resonance Spectroscopic Signals [1]

Diana M. Sima[2] Sabine Van Huffel[2]

June 2005

[1]This report is available by anonymous ftp from *ftp.esat.kuleuven.ac.be* in the directory *pub/sista/dsima/reports/SISTA5-120.pdf*

[2]ESAT-SISTA, K.U. Leuven, Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium, (diana.sima,sabine.vanhuffel@esat.kuleuven.be). This work was supported by **Research Council KUL:** GOA-AMBioRICS, several PhD/postdoc & fellow grants; **Flemish Government: FWO:** PhD/postdoc grants, projects, G.0407.02 (support vector machines), G.0269.02 (magnetic resonance spectroscopic imaging), G.0270.02 (nonlinear Lp approximation), G.0360.05 (EEG, Epileptic) research communities (ICCoS, ANMMM); **IWT:** PhD Grants, **Belgian Federal Science Policy Office:** IUAP P5/22 ('Dynamical Systems and Control: Computation, Identification and Modelling'); **EU:** PDT-COIL(NNE5/2001/887), BIOPATTERN (FP6-2002-IST 508803), ETUMOUR (FP6-2002-LIFESCIHEALTH 503094).

Abstract

This document describes algorithmic details and the FORTRAN 77 implementation of AQSES_{VP}, a software for quantification of short-echo time magnetic resonance spectroscopic signals. The quantification problem is mathematically formulated as a separable nonlinear least squares fitting problem. The detailed problem formulation, as well as the way that the Variable Projection technique can be applied to this problem, are described in the first half of the paper.

The FORTRAN 77 software implementation is discussed in the second half of the paper. This part is mainly meant as a basis for understanding the code, for future developments of this software package.

$\mathbf{AQSES_{VP}}$

Description of a Variable Projection Implementation for Nonlinear Least Squares with Linear Bounds Constraints, Applied to Accurate Quantification of Short-echo Time Magnetic Resonance Spectroscopic Signals

Diana M. Sima Sabine Van Huffel*

June 22, 2005

Abstract

This document describes algorithmic details and the FORTRAN 77 implementation of AQSES_{VP}, a software for quantification of short-echo time magnetic resonance spectroscopic signals. The quantification problem is mathematically formulated as a separable nonlinear least squares fitting problem. The detailed problem formulation, as well as the way that the Variable Projection technique can be applied to this problem, are described in the first half of the paper.

The FORTRAN 77 software implementation is discussed in the second half of the paper. This part is mainly meant as a basis for understanding the code, for future developments of this software package.

1 Introduction

1.1 The setting

The problem of quantifying metabolite concentrations from short-echo time *in vivo* NMR measurements [13, 11], such as measured proton spectrum from the human brain, is an important step towards correct non-invasive diagnosis of tumors. Each metabolite (chemical substance) has a peculiar time response in an NMR experiment; this time response has, in theory, the shape of a sum of complex damped exponentials. Spectra of metabolites that are relevant in the human brain can be measured *in vitro*. Such measurements can be grouped together in a database of metabolite signals.

A signal measured *in vivo* can be modeled as a combination of metabolites in the database plus a *baseline* signal that accounts for the presence of some unknown macromolecules. The quantities of interest are the *concentrations* of the metabolites, which can be computed from the weighting coefficients (amplitudes) in the combination of metabolites. However, in this combination one should allow small corrections in other spectral parameters (frequency shifts, damping corrections, phase shifts, etc), since these parameters may vary from measurement to measurement [15].

*ESAT-SCD, Department of Electrical Engineering, Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, 3001 Leuven, Belgium. E-mails: diana.sima@esat.kuleuven.be, sabine.vanhuffel@esat.kuleuven.be.

The baseline is a smooth, broadband, low amplitude spectrum, that gives an underlying trend to the *in vivo* signal, when visualized in the frequency domain. Its shape can vary and it is in general unpredictable, especially in pathological cases. For this reason, a good choice is found in identifying it as a smooth curve using nonparametric modeling.

Other important contributions in the field of fitting short-echo time NMR signals include [13] and [14]. The differences and similarities between the technique implemented in AQSES_{VP} and the procedure used in LCModel [13] are:

- AQSES_{VP} performs the fitting in the time-domain, which is the data acquisition domain; LCModel fits the real part of the frequency domain signals;
- in both methods, the baseline is nonparametrically modeled using penalized splines in the frequency domain; the fact that AQSES_{VP} fits in the time-domain is not a problem, because a smoothing criterion that involves an inverse Fourier transformed spline basis can be used;
- the criterion for choosing the amount of smoothing via the regularization parameter in AQSES_{VP} is based on nonlinear model selection theory, while the method in [13] seems more heuristic in nature.

Comparing AQSES_{VP} to the recent contribution in [14] (QUEST), we highlight the following differences:

- a nonparametric baseline is recovered in QUEST using heuristic methods, where several steps are involved: truncation, partial fitting, subtraction, and final fitting. Its performance is sensitive to the choice of the number of truncated data points and the model order for the baseline fit. The algorithm in AQSES_{VP} uses only one common optimization problem for the fitting of both the model and the baseline. It is thus less prone to accumulated errors.
- In reference [14], an augmented Fisher information matrix (inspired by [17]) is used, but it has a drawback: it is not clear how to choose the value for the number of effective parameters, involved in the computation of confidence bounds. In this respect, the discussion in [16] clarifies the way the confidence bounds can be automatically estimated for the procedure in AQSES_{VP} .

1.2 Mathematical formulation

For the quantification of short echo-time NMR signals, we consider that we are given a "metabolite database", which is a set $\{v_k, \text{ for } k = 1, \ldots, K\}$ of complex-valued time series of length m, representing *in vitro* measured NMR responses. Another complex-valued time series of length m, an *in vivo* measured NMR signal y, will satisfy the model

$$y(t) = \widehat{y}(t) + \varepsilon_t := \sum_{k=1}^{K} \alpha_k \, \zeta_k^t \, \eta_k^{t^2} \, v_k(t) + b(t) + \varepsilon_t, \quad t = t_0, \ldots, t_{m-1},$$

where $\alpha_k, \zeta_k, \eta_k \in \mathbb{C}$ are unknown parameters that account for concentrations of the metabolites in the database and for the necessary corrections of the database signals, due to inherent differences in the acquiring technique. In fact the complex amplitudes α_k and the complex ζ_k and η_k can be

written as (with $j = \sqrt{-1}$):

$$\begin{aligned}
\alpha_k &= a_k \exp(j\phi_k), \\
\zeta_k &= \begin{cases} \exp(-d_k + jf_k), & \text{for Lorentzian and Voigt lineshapes;} \\ \exp(jf_k), & \text{for Gaussian lineshapes;} \end{cases} \\
\eta_k &= \begin{cases} \exp(je_k), & \text{for Lorentzian lineshapes;} \\ \exp(-g_k + je_k), & \text{for Gaussian and Voigt lineshapes;} \end{cases}
\end{aligned}$$

where a_k are the real amplitudes, ϕ_k are the phase shifts, d_k are damping corrections, g_k are Gaussian damping corrections, f_k are frequency shifts, and e_k are Eddy current correction terms.

Moreover, $b(t)$ represents the chemical part that is not modeled, which is the response of the substances that are not included in the database, and ε_t is an unknown noise perturbation with zero mean, for all t's from 0 to $m-1$.

The identification of complex amplitudes α_k, and complex ζ_k's and η_k's, for $k = 1, \ldots, K$, can be accomplished by minimizing the least squares criterion: $\sum_{t=t_0,\ldots,t_{m-1}} |y(t) - \widehat{y}(t)|^2$.

The nuisance of incorporating the "baseline" term b is milded by the assumptions that can be made about it. Physiologically, the "macromolecules" (the chemicals that are left out of the database and are not relevant for the quantification) have some spectral characteristics in the NMR experiment, *e.g.*, they are broadband, low amplitude signals. In mathematical terms, we can say that b is characterized by the fact that its Fourier transformation should be a smooth function[1].

A semiparametric model with smoothness constraint for NMR quantification can be designed.

We construct a basis of splines [3, 4] and put the discretized splines as columns in a matrix A. Any nonlinear function can be approximated as a linear combination of spline functions. The coefficients in this linear combination are the unknowns that must be identified. We denote these linear coefficients by $a_1, \ldots, a_n$ (or by $\mathbf{a} \in \mathbb{C}^n$, when stacked in a column vector). Thus the discretization of a nonlinear function approximated with splines can be written in matrix notation as $A\mathbf{a}$.

A regularization operator D (that gives a matrix $B = D^\top D$) is defined to measure the smoothness of the baseline in the frequency domain. We can take D as the discrete second-order differential operator, $D = \begin{bmatrix} -1 & 2 & -1 & & 0 \\ & \ddots & \ddots & \ddots & \\ 0 & & -1 & 2 & -1 \end{bmatrix}$, first-order differential operator, $D = \begin{bmatrix} -1 & 1 & & 0 \\ & \ddots & \ddots & \\ 0 & & -1 & 1 \end{bmatrix}$, zero-order differential operator, $D = \begin{bmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{bmatrix}$, or some combination.

Since the goal is to reconstruct a smooth baseline in the frequency domain, while still fitting in the time-domain, we use back-transformation of the basis matrix A to the time domain, with the discrete inverse Fourier transform. Thus $\mathcal{A} := \mathcal{F}^{-1}(A)$, where the operator $\mathcal{F}^{-1}$ denotes the discrete inverse Fourier transform, applied on each column of a matrix. We consider the regularized nonlinear least squares criterion

$$\min_{(\alpha_1,\ldots,\alpha_K,\zeta_1,\ldots,\zeta_K,\eta_1,\ldots,\eta_K)\in\Omega,\ \mathbf{a}\in\mathbb{C}^n} \quad \frac{1}{m}\sum_{t=t_0}^{t_{m-1}} \left| y(t) - \sum_{k=1}^{K} \alpha_k\, \zeta_k^t\, \eta_k^{t^2}\, v_k(t) - (\mathcal{A}\mathbf{a})(t) \right|^2 + \lambda^2 \mathbf{a}^H B \mathbf{a}, \qquad (1)$$

where Ω denotes some constrained parameter space (which usually involves only linear equality constraints on the individual parameters). In (1), λ is a fixed regularization (penalty) parameter,

[1]There is an abuse of terms here: actually, the Fourier transform of the unknown *continuous-time* b should be a smooth function. In practice, we work with discrete time instants and with the discrete Fourier transform.

and the whole term $\lambda^2 \mathbf{a}^H B \mathbf{a}$ is responsible for ensuring a certain degree of smoothness to the baseline b. The value that we give to λ controls also the degree of smoothness.

Formulation (1) emphasises the fact that a baseline is fitted, while the original signal is used as given. However, another more general, and in some cases, more useful formulation is possible, which fits *filtered* signals to *filtered* models. We postpone the discussion upon this generalization to subsection 2.3.

2 Algorithmic details

2.1 When no baseline is fitted

The nonlinear least squares problem formulation (1) simply becomes

$$\min_{(\alpha_1,\ldots,\alpha_K,\zeta_1,\ldots,\zeta_K,\eta_1,\ldots,\eta_K)\in\Omega} \quad \frac{1}{m}\sum_{t=t_0}^{t_{m-1}} \left| y(t) - \sum_{k=1}^{K} \alpha_k\, \zeta_k^t\, \eta_k^{t^2}\, v_k(t) \right|^2 . \tag{2}$$

Problem (2) is a *separable* problem, where linear parameters α_k can be projected out of the least squares problem, and only a smaller sized nonlinear least squares problem remains to be solved for the nonlinear variables ζ_k, η_k, via an iterative minimization algorithm of the Levenberg-Marquardt type [12]. In what follows, we don't describe any details of the optimization solver; we take it as a black box solver that provides as output the optimal solution of the nonlinear least squares problem (2), and it requires initial values for the nonlinear parameters (we set initial values to zero, which means that we start the optimization with no spectral corrections to the signals in the database), linear bounds on each parameter, and procedures to evaluate the function value and the corresponding Jacobian, at each arbitrary set of parameter values.

2.1.1 Function evaluation

We rewrite (2) as

$$\min_{(\boldsymbol{\alpha},\boldsymbol{\zeta},\boldsymbol{\eta})\in\Omega} \frac{1}{m} \left\| \mathbf{y} - \Phi(\boldsymbol{\zeta},\boldsymbol{\eta})\boldsymbol{\alpha} \right\|_2^2 , \tag{3}$$

where $\mathbf{y}$ is a column vector containing $y(t_0),\ldots,y(t_{m-1})$, $\boldsymbol{\alpha}$, $\boldsymbol{\zeta}$ and $\boldsymbol{\eta}$ are defined as K-dimensional column vectors from the respective variables, and the $m \times K$ matrix $\Phi(\boldsymbol{\zeta},\boldsymbol{\eta})$ has elements of the form:

$$\Phi_{ik} = \zeta_k^{t_i} \eta_k^{t_i^2} v_k(t_i) = \begin{cases} \exp\left((-d_k + jf_k)t_i + je_k t_i^2\right)\ v_k(t_i), & \text{for Lorentzian lineshapes;} \\ \exp\left(jf_k t_i + (-g_k + je_k)t_i^2\right)\ v_k(t_i), & \text{for Gaussian lineshapes;} \\ \exp\left((-d_k + jf_k)t_i + (-g_k + je_k)t_i^2\right)\ v_k(t_i), & \text{for Voigt lineshapes;} \end{cases} \tag{4}$$

The optimal linear coefficients $\boldsymbol{\alpha}^{\text{opt}}$, for some fixed values of the nonlinear coefficients, $\boldsymbol{\zeta}$ and $\boldsymbol{\eta}$, satisfy[2]

$$\left\| \mathbf{y} - \Phi\boldsymbol{\alpha}^{\text{opt}} \right\|_2 = \min_{\boldsymbol{\alpha}\in\mathbb{C}^K} \left\| y - \Phi\boldsymbol{\alpha} \right\|_2,$$

and has the closed-form expression

$$\boldsymbol{\alpha}^{\text{opt}} = \Phi^{\dagger}\mathbf{y},$$

where $\Phi^{\dagger}$ denotes the pseudoinverse of the matrix Φ.

[2]We drop the dependence on the nonlinear parameters from the notation.

This implies that the residual we need to compute can be written as

$$\mathbf{y} - \Phi\boldsymbol{\alpha}^{\text{opt}} = (I - \Phi\Phi^{\dagger})\mathbf{y}.$$

Clearly, we only need a basis for the column space of the matrix Φ in order to evaluate the projection matrix $I - \Phi\Phi^{\dagger}$. This basis can be obtained from the QR decomposition of Φ:

$$\Phi = QR = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_1 \\ 0 \end{bmatrix},$$

where $R_1 \in \mathbb{C}^{K\times K}$, $Q_1 \in \mathbb{C}^{m\times K}$, and $Q_2 \in \mathbb{C}^{m\times(m-K)}$. Then, the residual $(I - \Phi\Phi^{\dagger})\mathbf{y}$ becomes $Q_2Q_2^H\mathbf{y}$. Since only the norm of this residual is in fact needed in the optimization algorithm, we can further simplify the definition of our residual by ignoring the multiplication with Q_2 (which has orthonormal columns), and simply compute $Q_2^H\mathbf{y}$ at each function evaluation.

2.1.2 Jacobian evaluation

The gradient of the residual $Q_2^H\mathbf{y}$ is also needed by the Levenberg-Marquardt type nonlinear least squares solver. Note that in the residual $Q_2^H\mathbf{y}$, the nonlinear parameters appear implicitly through Q_2.

Consider the original variable projection functional $f(x) = (I - \Phi(x)\Phi(x)^{\dagger})\mathbf{y}$, where the vector x stands for the nonlinear variables $\boldsymbol{\zeta}$, $\boldsymbol{\eta}$ (or, in full length, for the real nonlinear variables d, f, g, e). The Jacobian of f with respect to any of the scalar variables x_k (here denoted $\nabla_k f$) can be derived with the following manipulations:

$$\begin{aligned}
\nabla_k f &= -(\nabla_k\Phi)\Phi^{\dagger}\mathbf{y} - \Phi(\nabla_k(\Phi^{\dagger}))\mathbf{y} = -(\nabla_k\Phi)\Phi^{\dagger}\mathbf{y} - \Phi\nabla_k\left(\left(\Phi^H\Phi\right)^{-1}\Phi^H\right)\mathbf{y} \\
&= -(\nabla_k\Phi)\Phi^{\dagger}\mathbf{y} + \Phi\left(\Phi^H\Phi\right)^{-1}\left[(\nabla_k\Phi)^H\Phi + \Phi^H(\nabla_k\Phi)\right]\left(\Phi^H\Phi\right)^{-1}\Phi^H\mathbf{y} \\
&= -\left(\nabla_k\Phi - (\Phi^{\dagger})^H(\nabla_k\Phi)^H\Phi - \Phi\Phi^{\dagger}(\nabla_k\Phi)\right)\Phi^{\dagger}\mathbf{y}.
\end{aligned}$$

Kaufman's simplification [10] proposes that only the part

$$-\left(\nabla_k\Phi - \Phi\Phi^{\dagger}(\nabla_k\Phi)\right)\Phi^{\dagger}\mathbf{y} = -(I - \Phi\Phi^{\dagger})(\nabla_k\Phi)\Phi^{\dagger}\mathbf{y}$$

should be used to compute an approximate Jacobian, yielding a computational saving that is more important than the loss of accuracy in the Jacobian, which is negligible.

When we plug in the definition of Φ in (4), the matrix $\nabla_k\Phi$ can be computed using the formulas:

$$\frac{\partial\Phi_{ik}}{\partial d_k} = -t_i\Phi_{ik}, \qquad \frac{\partial\Phi_{ik}}{\partial f_k} = jt_i\Phi_{ik}, \qquad \frac{\partial\Phi_{ik}}{\partial g_k} = -t_i^2\Phi_{ik}, \qquad \frac{\partial\Phi_{ik}}{\partial e_k} = jt_i^2\Phi_{ik}. \tag{5}$$

Note here that the variable d_k (or f_k, or g_k, or e_k) only appears in the column k of Φ. Therefore, all other columns of $\nabla_k\Phi$ different from the column k are identically zero:

$$\frac{\partial\Phi_{il}}{\partial d_k} = 0, \qquad \frac{\partial\Phi_{il}}{\partial f_k} = 0, \qquad \frac{\partial\Phi_{il}}{\partial g_k} = 0, \qquad \frac{\partial\Phi_{il}}{\partial e_k} = 0, \qquad \text{for } l \neq k.$$

It is possible to require also the estimation of the initial time instant t_0, when using the convention that each t_i equals $t_0 + i\Delta t$, for a given time step $\Delta t > 0$. Then,

$$\frac{\partial\Phi_{ik}}{\partial t_0} = \begin{cases} (-d_k + jf_k + je_kt_i)\Phi_{ik}, & \text{for Lorentzian lineshapes;} \\ (jf_k + (-g_k + je_k)t_i)\Phi_{ik}, & \text{for Gaussian lineshapes;} \\ (-d_k + jf_k + (-g_k + je_k)t_i)\Phi_{ik}, & \text{for Voigt lineshapes.} \end{cases}$$

To fix the ideas, the matrix $\nabla_k \Phi$, which represents the derivative of the matrix Φ with respect to the k^{th} variable x_k (that is either d_k, or f_k, or g_k, or e_k), is an $m \times K$ matrix that has the following structure

$$\nabla_k \Phi = \begin{bmatrix} 0 & 0 & \dfrac{\partial \Phi_{1k}}{\partial x_k} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dfrac{\partial \Phi_{mk}}{\partial x_k} & 0 & 0 \end{bmatrix}$$

Thus, the column corresponding to x_k in the approximate Jacobian equals

$$\widetilde{\nabla}_k f = -(I - \Phi\Phi^\dagger)(\nabla_k \Phi)\Phi^\dagger \mathbf{y} = -(I - \Phi\Phi^\dagger) \begin{bmatrix} \dfrac{\partial \Phi_{1k}}{\partial x_k} \\ \vdots \\ \dfrac{\partial \Phi_{mk}}{\partial x_k} \end{bmatrix} \boldsymbol{\alpha}_k^{\text{opt}}, \tag{6}$$

where we used the fact that $\boldsymbol{\alpha}^{\text{opt}} = \Phi^\dagger \mathbf{y}$. The complete approximate Jacobian $\widetilde{\nabla} f$ is obtained by putting next to each other all columns of type (6), one column for each nonlinear variable in our optimization.

For stable and efficient computation of the Jacobian, we make use of the QR decomposition of Φ, as introduced before. Thus, $\boldsymbol{\alpha}^{\text{opt}} = R_1^{-1} Q_1^H \mathbf{y}$ and $I - \Phi\Phi^\dagger = Q_2 Q_2^H$. Since we ignore the factor Q_2 in the function evaluation, we may also do the same thing in what concerns the Jacobian.

In the end, the approximate Jacobian using Kaufman's simplification is:

$$\widetilde{\nabla} f = -Q_2^H \Delta\Phi, \tag{7}$$

where $\Delta\Phi$, in its most general formulation and with k going from 1 to K, is

$$\left[\begin{array}{ccc|ccc|ccc|ccc|ccc} \cdot & \boldsymbol{\alpha}_k^{\text{opt}} \dfrac{\partial \Phi_{1k}}{\partial d_k} & \cdot & \cdot & \boldsymbol{\alpha}_k^{\text{opt}} \dfrac{\partial \Phi_{1k}}{\partial g_k} & \cdot & \cdot & \boldsymbol{\alpha}_k^{\text{opt}} \dfrac{\partial \Phi_{1k}}{\partial f_k} & \cdot & \cdot & \boldsymbol{\alpha}_k^{\text{opt}} \dfrac{\partial \Phi_{1k}}{\partial e_k} & \cdot & \cdot & \dfrac{\partial \Phi_{1k}}{\partial t_0} & \cdot \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdot & \boldsymbol{\alpha}_k^{\text{opt}} \dfrac{\partial \Phi_{mk}}{\partial d_k} & \cdot & \cdot & \boldsymbol{\alpha}_k^{\text{opt}} \dfrac{\partial \Phi_{mk}}{\partial g_k} & \cdot & \cdot & \boldsymbol{\alpha}_k^{\text{opt}} \dfrac{\partial \Phi_{mk}}{\partial f_k} & \cdot & \cdot & \boldsymbol{\alpha}_k^{\text{opt}} \dfrac{\partial \Phi_{mk}}{\partial e_k} & \cdot & \cdot & \dfrac{\partial \Phi_{mk}}{\partial t_0} & \cdot \end{array}\right]$$

In cases when not all variables among d_k, g_k, f_k, e_k, t_0 are estimated in the model, or when we impose *prior knowledge* in the form of linear equalities between some variables of the same sort (leading to eliminations), the formula above simplifies by deleting the not-needed columns.

2.2 When a baseline is also fitted

Little is changed in the variable projection implementation, when we augment the optimization criterion (2) to the regularized version (1). In fact, using the notation from (3), the minimization (1) can be written as

$$\min_{(\boldsymbol{\alpha},\boldsymbol{\zeta},\boldsymbol{\eta})\in\Omega,\, \mathbf{a}\in\mathbb{C}^n} \frac{1}{m} \left\| \begin{bmatrix} \mathbf{y} \\ 0 \end{bmatrix} - \begin{bmatrix} \Phi(\boldsymbol{\zeta},\boldsymbol{\eta})\boldsymbol{\alpha} + \mathcal{A}\mathbf{a} \\ \sqrt{m}\lambda D\mathbf{a} \end{bmatrix} \right\|_2^2,$$

which is also a separable nonlinear least squares problem, where the linear variables are $\boldsymbol{\alpha}$ and $\mathbf{a}$, and the nonlinear ones are $\boldsymbol{\zeta}$ and $\boldsymbol{\eta}$.

For the function evaluation, we use the QR decomposition of the matrix $\begin{bmatrix} \mathcal{A} & \Phi(\boldsymbol{\zeta},\boldsymbol{\eta}) \\ \sqrt{m}\lambda D & 0 \end{bmatrix} = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$, with $R_1 \in \mathbb{C}^{(n+K)\times(n+K)}$, and Q_1, Q_2 of appropriate sizes. The function value

is $Q_2^H \begin{bmatrix} \mathbf{y} \\ 0 \end{bmatrix}$, and the projected linear variables have the expression

$$\begin{bmatrix} \boldsymbol{\alpha}^{\text{opt}} \\ \mathbf{a}^{\text{opt}} \end{bmatrix} = R_1^{-1} Q_1^H \begin{bmatrix} \mathbf{y} \\ 0 \end{bmatrix}.$$

For approximate Jacobian evaluation, the only difference comes from the fact that we have augmented $\Phi(\boldsymbol{\zeta}, \boldsymbol{\eta})$ with some blocks that *do not* depend on the nonlinear parameters. This translates into the fact that the new Jacobian is also extended with zero blocks of corresponding dimensions. All completely zero columns will be ignored in the implementation.

2.3 Using a filter

A filter can be used to remove irrelevant information from the *in vivo* NMR signal. For instance, a pass-band FIR filter can be used to select only the frequency region of interest. The FIR filter in AQSES$_{\text{VP}}$ is taken from the implementation of [18]; it is a maximum phase FIR filter that is automatically optimized in order to remove the water component from an NMR *in vivo* signal.

Applying a FIR filter to a vector (a discrete signal) involves a convolution operation. This is however a simple and fast computation. When such an operation is applied to the measured signal, it must also be taken into account by the fitting model. In other words, a filtered measured signal will be fitted with a filtered model plus a filtered baseline.

The design of the filter is performed outside the actual fitting method of AQSES$_{\text{VP}}$. Such a filter consists of a vector of coefficients; the length of the filter and the coefficients are optimized during the automatic filter design. Let $h_1, \ldots, h_p$ denote the filter coefficients. Then a discrete-time signal v, *i.e.*, a vector of length m, can be convolved with the filter giving the filtered signal w (of length $m - p + 1$), according to

$$w_i = \sum_{l=1}^{p} h_l v_{i+l-1}, \qquad i = 1, \ldots, m - p + 1.$$

A FIR filter commutes with the sum, but this doesn't mean that it commutes also with the *modified* sum of metabolites, where there are shifts and corrections to the spectral parameters; thus we cannot apply the same methodology as for unfiltered signals, only by using a filtered database. In AQSES$_{\text{VP}}$'s minimization, we replace the original signal, the reconstructed signal and the reconstructed baseline with their filtered versions. To be more precise, the changes that are involved in the function and Jacobian evaluation are only the following three:

- the *in vivo* signal y is replaced throughout with its filtered version;
- the matrix Φ constructed in (4) will be replaced at each function (and Jacobian) evaluation with a matrix whose columns are filtered versions of the columns of the original Φ;
- the spline matrix $\mathcal{A}$ is replaced throughout with its filtered version (each column is thus separately filtered).

3 Implementation details

3.1 Programming language and dependencies

AQSES$_{\text{VP}}$ is implemented in FORTRAN 77. The main optimization part is carried out using an extension of the Levenberg-Marquardt algorithm, which accepts linear bounds constraints, in the

DN2GB implementation written by David M. Gay [6]. DN2GB is an extension of the NL2SOL package of the same author [9], which deals with nonlinear least squares minimization for real data. AQSES$_{\mathrm{VP}}$ is performing nonlinear least squares with a complex-valued function, *i.e.*, a complex norm

$$\|f(\theta)\|^2 = \|\Re f(\theta)\|_2^2 + \|\Im f(\theta)\|_2^2$$

is minimized, where $f(\theta) = \Re f(\theta) + i\Im f(\theta) \in \mathbb{C}^m$. Thus, instead of solving an optimization problem with complex data, it is possible to transform all computations to real by doubling the problem dimension.

We have chosen for a reimplementation of the Variable Projection method [7, 8], taking the guidelines from the VARPRO implementation written in FORTRAN by John Bolstad [19]. Our new code for nonlinear least squares with variable projection embedded in AQSES$_{\mathrm{VP}}$ works directly with complex function computations. That is, function and Jacobian evaluations are performed in the complex domain. The evaluated values are transformed to real in order to apply the optimization module DN2GB. Note that in this way we are able to impose linear bounds on the nonlinear parameters involved in the minimization, a fact not implemented by the classical VARPRO code.

AQSES$_{\mathrm{VP}}$ uses several routines from BLAS and LAPACK [1] for some basic linear algebra computation with (double) real as well as (double) complex data. A few routines from SLICOT [2] and from the FFTW package [5] are also used.

3.2 Multiple round fitting

AQSES$_{\mathrm{VP}}$ is designed with an option of multiple round minimization: fitting a new signal with the signals from a database can be done in one or several steps. The order of the components in the database (where each column corresponds to a certain discretized time-domain signal) is important. An input variable `nrounds` gives the number of rounds in the optimization, and an input vector `landmarks` of length `nrounds` gives the number of components from the database that are used in each of the rounds. That is, the first `landmarks(1)` columns of `database` are used for fitting in the first round, and so on, till the last round when the first `landmarks(nrounds)` columns of `database` are used in the fitting. Note that in this way there might be database components left out from the fitting, even in the last round.

3.3 Testing the validity of databases

Two tests are provided in the code in order to test the numerical validity of the database and to warn if numerical sensitivity of the database might influence the correctness of the fitting results. Both tests act in the same way, but the first one is done for the original database, while the second is done for the filtered database, in case that a filter is used.

Numerical problems during the nonlinear least squares fitting procedure might occur in the case when there exist nearly linearly dependent columns in the (filtered) database. In this situation, the Jacobian computed for each step of the Levenberg-Marquardt minimization algorithm gets nearly rank deficient and causes computational problems.[3]

[3]It is true that Levenberg-Marquardt has guards against rank-deficient Jacobian, but this typical rank-deficiency is usually caused in nonlinear least squares minimization by nearly compatible nonlinear equation fitting. The case of linearly dependent columns in the database is a degenerate situation yielding rank-deficient Jacobian, and should be avoided.

3.4 Code structure

3.4.1 Driver routine: `drivervp.f`

The `drivervp.f` function has the role of coordinating the main computational flow. The steps performed by `drivervp.f` are:

- initializations
- filtering, if required
- for each round, except last round
  - fit signal using current database (see `mainround.f`)
- in the last round
  - fit signal using current database (see `mainround.f`) and baseline if required (see `setsplines.f` and `initsvspace.f`)
  - optimize baseline smoothness parameter by performing previous step repeatedly and evaluating Generalized Cross-Validation function
- compute estimated error bounds (see `crerrorvp.f`)

3.4.2 Computations for each round: `mainround.f`

Function `mainround.f` performs the whole optimization process of a single round and deals with both cases of considering a baseline in the fit or not). The steps follow:

1. initialize optimization parameters (`inivp.f`):
   - put all free (nonlinear) variables in the vector `alf`, in the order: dampings, Gaussian factors, frequencies, Eddy current factors;
   - count the number of variables and set `nrvar` to this value;
   - set corresponding linear bounds (that were given in the input variables ending in `-low` and `-upp`, *e.g.*, `freqlow` and `frequpp`) for all variables into a $2\times$ `nrvar` matrix of lower and upper bounds `B`.
2. compute number of linear parameters, number of constraints, and some index values for referencing different blocks within the complex workspace
3. if baseline is used, then the baseline spline coefficient matrix and the smoothness penalty matrix should be copied to the "saved spaced", operation done in `initsvspace.f`. The part containing the penalty matrix $\sqrt{m}\lambda D$ is updated in `updsvspace.f` by using the newest value of the smoothness parameter `lambda`. The call to `updsvspace.f` provides also the QR decomposition of the blocks containing the baseline coefficient matrix and penalty matrix. This way, the QR factorization of the matrix $\begin{bmatrix} \mathcal{A} & \Phi(\boldsymbol{\zeta},\boldsymbol{\eta}) \\ \sqrt{m}\lambda D & 0 \end{bmatrix}$ that is needed at each function and Jacobian evaluations will be updated only in the block column that contains $\Phi(\boldsymbol{\zeta},\boldsymbol{\eta})$, thus saving computations.

4. main optimization takes place, using the (slightly modified) `DN2GB` module `dn2gbvp.f`. Function and Jacobian evaluations are made through the `funcvp.f` and `jacovp.f` functions, where all operations are performed with complex data (*e.g.*, evaluation of the linear parameters), but only at the end the complex data is transformed to real and fed to the `DN2GB` optimization solver. More details in § 3.4.3 and § 3.4.4.

5. optimal nonlinear parameters stored in `alf` are transformed back to meaningful spectral variables `damp`, `gauss`, `freq`, `eddy`, `t0` in the function `updatevp.f`.

6. optimal linear parameters corresponding to the optimal nonlinear variables in `alf` and computed as linear least squares solution are transformed to meaningful real spectral parameters `ampl` and `phas`, and to the baseline (if computed), in `updatelin.f`.

3.4.3 Function evaluation: `funcvp.f`

The following steps are taken in `funcvp.f`:

- The matrix Φ from (4) is efficiently computed, by first evaluating the complex exponentials for each index k, but for a fixed value of $t_i = 1$, and then filling in each row of Φ with appropriate values for each t_i. This means that instead of computing mK complex exponentials, we compute only K complex exponentials and we raise $2mK$ complex numbers to real powers.

  The matrix Φ is stored in the array `expon` and it is kept for possible use in the Jacobian evaluation. We rely on the fact that there is always a function evaluation before any Jacobian evaluation at the current parameter values.

- In the case when a FIR filter is required, each column of Φ is filtered.

- The filtered version of Φ is stored in the reserved block of `svSpace`, which is either the upper `nrpf` $\times$ `effbasis`, when no baseline is used, or in the upper right block next to the filtered spline matrix `filM`, when also the baseline is used. Thus, the `svSpace` workspace contains in fact the matrix $\begin{bmatrix} \mathcal{A} & \Phi(\boldsymbol{\zeta}, \boldsymbol{\eta}) \\ \sqrt{m}\lambda D & 0 \end{bmatrix}$, but with filtered columns corresponding to $\mathcal{A}$ and $\Phi(\boldsymbol{\zeta}, \boldsymbol{\eta})$, and with a partial QR factorization in the left block column.

- The complete QR factorization of `svSpace` is computed, and a compact representation is saved in the subdiagonal part of the first `nrlinpar` columns of `svSpace`, and in the vector `tau`.

- The residual $Q^H \mathbf{y}$ is computed and stored in the (`nrlinpar` $+ 1)^{\text{st}}$ column of `svSpace`. Then, this complex residual is transformed to real, by unfolding real and imaginary parts into odd and even locations of the vector `fvecf`.

3.4.4 Jacobian evaluation: `jacovp.f`

$\nabla_k \Phi$ is stored in a compact way, by keeping only nonzero columns. In other words, if a certain column of Φ does not depend on a certain variable x_k, then the all-zero column will not be stored.

The following steps are taken in `funcvp.f`:

- Linear variables are computed from the linear least squares problem that involves the current Φ. Since we already have the QR decomposition of the coefficient matrix in the least squares problem, the complex vector of linear parameters is easily computed by solving a triangular linear system of equations. (The same computation is performed in the routine

`updatelin.f`, for extracting the linear parameters at the end of the whole optimization process.)

The linear coefficients are stored in the $(\texttt{nrlinpar}+1)^{\text{th}}$ column of `svSpace`, with the baseline coefficients in the first `nosplpars` positions (note that when the baseline is not required, `nosplpars` is zero), and the complex amplitudes in the next `effbasis` positions.

- For each of the active nonlinear variables (dampings, Gaussian dampings, frequencies, Eddy current corrections), a corresponding column is added to the Jacobian. The implemented formulas are found in Subsection 2.1.2.
- If a filter is required, then each stored column is also passed through the FIR filter, and the columns from `nrlinpar` + 2 to `nrlinpar` + `nrvar` + 2 of `svSpace` are used for storage.
- A zero block corresponding to the penalty part of the optimization is added under the $\Delta\Phi$ block, since the penalty term does not involve the nonlinear parameters.
- Kaufman's simplification is used; it implies that the above mentioned columns of `svSpace` are multiplied with the block $-Q_2$ of the unitary matrix Q from the already computed QR factorization (see (7)). Notice that Q_2 is compactly stored in the lower trapezoidal part of the first `nrlinpar` columns of `svSpace` and in the vector `tau`, according to LAPACK's ZGEQRF routine conventions.
- The complex approximate Jacobian is transformed to real, by unfolding real and imaginary parts into odd and even rows of the matrix `fjacf`.

3.5 Management of the complex workspace

3.5.1 Complex memory allocation

A short explanation that can also be found in the help of `drivervp.f` follows.

```
'workspace' is used in dn2gbvp.f (and dependencies, such as
funcvp.f, jacovp.f) to allocate memory for the following
arrays in those routines:

expon   = workspace(1), length nrvar*nrp
svSpace = workspace(indsvsp), length ldsv*(nrlinpar+nrvar+1)
          with ldsv >= nrconstr    and indsvsp = nrvar*nrp+1
M       = workspace(indM),    length LM*nosplpars
          with LM = nrp, nosplpars<nrp,
          indM = indsvsp + ldsv*(nrlinpar+nrvar+1)
filM    = workspace(indfilM), length ldsv*nosplpars,
          with indfilM = indM + LM*nosplpars
tau     = workspace(indtau),  length nrlinpar
          with  indtau = indfilM + ldsv*nosplpars
wkspace = workspace(indw),    length ldw >= nrp*nrvar
          with  indw   = indtau + nrlinpar

where:

nrlinpar = nrbasis + basetrue*nosplpars < nrbasis + basetrue*nrp
```

```
nrconstr = nrpf + basetrue*(nosplpars - sord)
         < nrp + basetrue*(nrp - 2) < 2*nrp
nosplpars< nrp, if basetrue = 1
nosplpars= 0,   if bastrue  = 0

Thus,

total_length_workspace = nrvar*nrp +
                         ldsv*(nrlinpar+nrvar+1) +
                         (LM+ldsv)*nosplpars +
                         nrlinpar +
                         nrp*nrvar
  ( if basetrue=1 )   <= 2*nrp*(4*nrbasis) +  3*nrp*nrp +
                         2*nrp*(nrbasis+nrp+4*nrbasis+1) +
                         nrbasis + nrp
                 = 5*nrp**2 + 18*nrp*nrbasis + 3*nrp + nrbasis

  ( if basetrue=0 )   <= 2*nrp*(4*nrbasis) +
                         nrp*(nrbasis+4*nrbasis+1) +
                         nrbasis
                 = 18*nrp*nrbasis + nrp + nrbasis
```

3.5.2 Workspace dimensions and pointers

Here is more explanation about the variables that hold various indices, number of variables, spline information, lengths of allocated memory:

`nrlinpar` is the number of linear parameters, thus the number of parameters that can be optimally computed via linear least squares, when fixed values are given for the nonlinear parameters.

`effbasis` is the current number of database components used in the fit; *e.g.*, in the second run, `effbasis` = `landmarks(2)`.

`basetrue` is a flag variable equal to 1 if a baseline is used in the fit, and to 0, otherwise.

`nosplpars` is the number of spline parameters, which is initialized in `setsplines.f`.

`nrconstr` is the number of linear constraints that define the linear parameters.
`nrconstr` and `nrlinpar` give the size of the coefficient matrix $\Phi(\texttt{alf})$, where `alf` is the current vector of nonlinear variables, and the current optimal linear variables z are given as the least squares solution of the system

$$\Phi(\texttt{alf})z \approx \mathbf{y} \qquad \Rightarrow \qquad z = \Phi(\texttt{alf})^{\dagger}\mathbf{y},$$

where $\mathbf{y}$ is the signal to be analyzed.

`nrpf` is the number of filtered data points.

`sord` is the degree of the smoothness constraint, with the following conventions:

```
= 0, penalty matrix = identity matrix
= 1, penalty matrix = first order derivative matrix
```

```
= 2, penalty matrix = second order derivative matrix
= 3, penalty matrix = identity matrix, followed by the
                      second order derivative matrix
```

`indsvsp` is the index in the complex `workspace` matrix where the "saved workspace" begins (see § 3.5.1).

`indtau` is the index in the complex `workspace` matrix where the vector `tau` from saved Householder rotations begins (see § 3.5.1).

`ldsv` is the leading dimension of the "saved workspace" (see § 3.5.1).

`indw` is the index in the complex `workspace` matrix where the remaining complex workspace begins. This extra workspace is needed by subordinate routines such as LAPACK's ZGEQRF routine for QR factorization (see § 3.5.1).

`ldw` is the length of the remaining complex workspace.

Acknowledgments

Diana Sima is a research assistant and Dr. Sabine Van Huffel is a full professor at the Katholieke Universiteit Leuven, Belgium. Research supported by

Research Council KUL: GOA-AMBioRICS, several PhD/postdoc & fellow grants;
Flemish Government: FWO: PhD/postdoc grants, projects, G.0407.02 (support vector machines), G.0269.02 (magnetic resonance spectroscopic imaging), G.0270.02 (nonlinear Lp approximation), G.0360.05 (EEG, Epileptic), research communities (ICCoS, ANMMM); **IWT:** PhD Grants;
Belgian Federal Science Policy Office: IUAP P5/22 ('Dynamical Systems and Control: Computation, Identification and Modelling');
EU: PDT-COIL, BIOPATTERN, ETUMOUR.

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorenson. *LAPACK Users' Guide, 2nd Edition.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1995.

[2] P. Benner, V. Mehrmann, V. Sima, S. Van Huffel, and A. Varga. Slicot – a subroutine library in systems and control theory. *Applied and Computational Control, Signals and Circuits*, pages 499–539, 1998.

[3] C. de Boor. *A Practical Guide to Splines.* Springer, Berlin, 1978.

[4] P. H. C. Eilers and B. D. Marx. Flexible smoothing with B-splines and penalties. *Statistical Science*, 11(2):89–121, 1997.

[5] M. Frigo and S. G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005. special issue on "Program Generation, Optimization, and Platform Adaptation".

[6] D. M. Gay. A trust-region approach to linearly constrained optimization. In D. F. Griffiths, editor, *Numerical Analysis Proceedings*, pages 72–105, Dundee, 1983. Springer-Verlag.

[7] G. H. Golub and V. Pereyra. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM Journal on Numerical Analysis*, 10:413–432, 1973.

[8] G. H. Golub and V. Pereyra. Separable nonlinear least squares: the variable projection method and its applications. *Inverse Problems*, 19(2):1–26, 2003.

[9] J. John E. Dennis, D. M. Gay, and R. E. Welsch. Algorithm 573: Nl2sol – adaptive nonlinear least-squares algorithm [e4]. *ACM Trans. Math. Softw.*, 7(3):369–383, 1981.

[10] L. Kaufman. A variable projection method for solving separable nonlinear least squares problems. *BIT*, 15:49–57, 1975.

[11] P. Lemmerling, L. Vanhamme, H. J. A. in't Zandt, S. Van Huffel, and P. Van Hecke. Time-domain quantification of short-echo-time in-vivo proton MRS. *MAGMA*, 15:178–179, 2002.

[12] J. J. Moré. The Levenberg-Marquardt algorithm: Implementation and theory. In G. A. Watson, editor, *Numerical Analysis: Proceedings of the Biennial Conference held at Dundee, June 28-July 1, 1977 (Lecture Notes in Mathematics #630)*, pages 104–116. Springer Verlag, 1978.

[13] S. W. Provencher. Estimation of metabolite concentrations from localized *in vivo* proton NMR spectra. *Magnetic Resonance in Medicine*, 30(6), 1993.

[14] H. Ratiney, Y. Coenradie, S. Cavassila, D. van Ormondt, and D. Graveron-Demilly. Time-domain quantitation of ^{1}H short echo-time signals: Background accommodation. *MAGMA*, 16:284–296, 2004.

[15] H. Ratiney, M. Sdika, Y. Coenradie, S. Cavassila, D. van Ormondt, and D. Graveron-Demilly. Time-domain semi-parametric estimation based on a metabolite basis set. *NMR in Biomedicine*, 17:1–13, 2004.

[16] D. M. Sima and S. Van Huffel. Regularized semiparametric model identification (with application to NMR signal quantification with unknown macromolecular baseline). Technical Report 04-229, K.U. Leuven, E.E. Dept. (ESAT-SISTA), Kasteelpark Arenberg 10, 3001 Leuven, Belgium, December 2004. Available from `ftp://ftp.esat.kuleuven.ac.be/pub/SISTA/dsima/abstracts/04-229.html`.

[17] J. C. Spall and J. P. Garner. Parameter identification of state-space models with nuisance parameters. *IEEE Transactions on Aerospace and Electronic Systems*, 26(6):992–998, 1990.

[18] T. Sundin, L. Vanhamme, P. Van Hecke, I. Dologlou, and S. Van Huffel. Accurate quantification of ^{1}H spectra: from FIR filter design for solvent suppression to parameter estimation. *Journal of Magnetic Resonance*, 139:189–204, 1999.

[19] `http://www.netlib.org/opt/varpro`.