IMM
DEPARTMENT OF MATHEMATICAL MODELLING

Technical University of Denmark
DK-2800 Lyngby – Denmark

J. No. MEXP
17.2.2000
HBN/ms

MULTI-EXPONENTIAL FITTING OF LOW-FIELD ^{1}H NMR DATA

Hans Bruun Nielsen

TECHNICAL REPORT

IMM-REP-2000-03

IMM

MULTI-EXPONENTIAL FITTING OF LOW-FIELD ^{1}H NMR DATA

Hans Bruun Nielsen

Contents

1. Introduction

A typical set of low-field ^{1}H NMR relaxation data $(t_i, y_i),\ i = 1, \ldots, m$, is shown below.

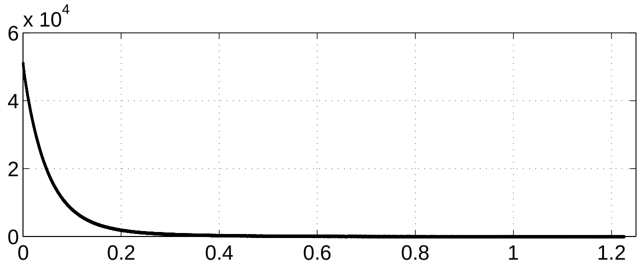

Figure 1.1. *Data points*

It can be modelled by

$$y_i = M(\mathbf{x}, \mathbf{c}, t_i) + \varepsilon_i \ , \tag{1.1a}$$

where

$$M(\mathbf{x}, \mathbf{c}, t) = c_0 + \sum_{j=1}^{n} c_j e^{x_j t} = \sum_{j=0}^{n} c_j e^{x_j t} \ , \tag{1.1b}$$

with $x_0 = 0$. The $\{\varepsilon_i\}$ are "white noise". satisfying

$$E(\varepsilon_i) = 0, \quad E(\varepsilon_i \varepsilon_j) = \delta_{ij} \varepsilon^2 \ , \tag{1.1c}$$

where δ_{ij} denotes the Kronecker delta. We shall use the term "noise level" to denote the standard deviation ε.

We want to estimate the parameters by means of least squares. If n is known and a good starting guess for the $\{x_j\}$ is available, we recommend to exploit the fact that the model is *separable*, see Section 5. In this report we discuss the common case, where we do not have this *a priori* knowledge, but we know that m is large, n is small $(n \leq 6)$ and the data abscissae are equidistant, $t_{i+1} \Leftrightarrow t_i = h$.

Our approach is based on methods for other types of NMR analysis, where it is recommended (see e.g. [2]) to exploit that the model satisfies a *linear predictor*,

$$M(\mathbf{x},\mathbf{c},t_i) = \sum_{k=1}^{K} f_k M(\mathbf{x},\mathbf{c},t_{i-k}) \tag{1.2a}$$

for any $K \geq n$ and $i \geq K$. This is a *forward* predictor. Alternatively, we can use a *backward* predictor

$$M(\mathbf{x},\mathbf{c},t_i) = \sum_{k=1}^{K} b_k M(\mathbf{x},\mathbf{c},t_{i+k}) \ . \tag{1.2b}$$

If we use the latter relation for $i=1,\ldots,q$ and replace $M(\mathbf{x},\mathbf{c},t_s)$ by y_s, we get the following system of equations,

$$\mathbf{H}_{:,2:K+1}\mathbf{b} \simeq \mathbf{H}_{:,1} \ , \tag{1.3a}$$

where $\mathbf{b} = [b_1,\ldots,b_K]^\top$ and $\mathbf{H}_{:,2:K+1}$ denotes the submatrix consisting of columns $2,\ldots,K+1$ in

$$\mathbf{H} = \begin{bmatrix} y_1 & y_2 & \cdots & y_{1+K} \\ y_2 & y_3 & \cdots & y_{2+K} \\ \vdots & \vdots & & \vdots \\ y_q & y_{q+1} & \cdots & y_{q+K} \end{bmatrix} \ . \tag{1.3b}$$

A similar derivation shows that the coefficients $\mathbf{f} = [f_K,\ldots,f_1]^\top$ of the forward predictor satisfy

$$\mathbf{H}_{:,1:K}\mathbf{f} \simeq \mathbf{H}_{:,K+1} \ . \tag{1.3c}$$

Note that q and K must satisfy

$$q + K \leq m \quad \Leftrightarrow \quad q \leq m \Leftrightarrow K \ . \tag{1.4}$$

Suppose that we take the largest possible value for q, $q = m \Leftrightarrow K$, and demand that $q > K \ \Leftrightarrow \ K < \frac{1}{2}m$. Then (1.3a,c) are overdetermined systems of equations. We take $\mathbf{b}$ (or $\mathbf{f}$) as a least squares solution and in Section 2 we discuss how this can be used to find the parameters $\mathbf{x}$ and $\mathbf{c}$. The behaviour of the algorithm is illustrated in Section 3, while enhancements are presented in Sections 4 – 6. Finally, in Sections 7 – 8 we apply the algorithm to the data of Figure 1.1 and attempt some conclusions.

2. Analysis

The assumption about equidistant data abscissae implies that

$$t_i = a + ih \ , \tag{2.1a}$$

and therefore

$$\begin{aligned} M(\mathbf{x},\mathbf{c},t_i) &= \sum_{j=0}^{n} c_j e^{x_j(a+ih)} \\ &= \sum_{j=0}^{n} c_j e^{x_j a} \cdot \left(e^{x_j h}\right)^i \equiv \sum_{j=0}^{n} \tilde{c}_j \xi_j^i \ . \end{aligned} \tag{2.1b}$$

In the following we shall not distinguish between c_j and $\tilde{c}_j$ (they are equal in the typical case, where $a=0$). Inserting this expression in (1.2b) we get

$$\sum_{j=0}^{n} c_j \xi_j^i = \sum_{k=1}^{K} b_k \Big(\sum_{j=0}^{n} c_j \xi_j^{i+k}\Big) = \sum_{j=0}^{n} c_j \xi_j^i \Big(\sum_{k=1}^{K} b_k \xi_j^k\Big) \ . \tag{2.2a}$$

This is satisfied for all $i = 1,\ldots m \Leftrightarrow K$ if the $\{\xi_j\}$ are roots of the polynomial

$$\widetilde{P}_K(\xi) = 1 \Leftrightarrow b_1\xi \Leftrightarrow \cdots \Leftrightarrow b_K \xi^K \ . \tag{2.2b}$$

Similarly we see that

$$\sum_{j=0}^{n} c_j\xi_j^i = \sum_{j=0}^{n} c_j\xi_j^i\Big(\sum_{k=1}^{K} f_k\xi_j^{-k}\Big) \quad , \tag{2.3a}$$

and this is satisfied for all $i = K{+}1, \dots m$ provided that the $\{\xi_j\}$ are roots of the polynomial

$$P_K(\xi) = \xi^K \Leftrightarrow f_1\xi^{K-1} \Leftrightarrow \cdots \Leftrightarrow f_K \ . \tag{2.3b}$$

Thus, having computed $\mathbf{b}$ (or $\mathbf{f}$) by means of (1.3), we get $x_j = (\ln\xi_j)/h,\ j = 1,\dots,n$, where the relevant ξ_j values are chosen among the K roots of $\widetilde{P}_K$ (or P_K). We return to this point later. In the special case where $c_0 = 0$ and $K = n$ the method based on the forward predictor is identical with *Prony's algorithm*, [12].

In order to relate to (1.3) we introduce

$$\mathbf{c} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} , \quad \boldsymbol{\xi}^i = \begin{bmatrix} 1 \\ \xi_1^i \\ \vdots \\ \xi_n^i \end{bmatrix} , \quad \boldsymbol{\Xi}^i = \mathrm{diag}(\boldsymbol{\xi}^i) \ . \tag{2.4}$$

Then we can write $M(\mathbf{x}, \mathbf{c}, t_i) = \mathbf{c}^\top\boldsymbol{\xi}^i$, and – in case of exact data,

$$\mathbf{H}_{:,j+1} = \begin{bmatrix} \mathbf{c}^\top\boldsymbol{\xi}^{j+1} \\ \vdots \\ \mathbf{c}^\top\boldsymbol{\xi}^{j+q} \end{bmatrix} = \begin{bmatrix} \mathbf{c}^\top\boldsymbol{\Xi}^1 \\ \vdots \\ \mathbf{c}^\top\boldsymbol{\Xi}^q \end{bmatrix} \boldsymbol{\xi}^j = \mathbf{Y}\boldsymbol{\xi}^j \ , \tag{2.5a}$$

or

$$\mathbf{H} = \mathbf{Y}\left[\boldsymbol{\xi}^0\ \boldsymbol{\xi}^1\ \cdots\ \boldsymbol{\xi}^K\right] = \mathbf{Y}\mathbf{G} \ . \tag{2.5b}$$

Assume that $q \geq K \geq n$. By nature the $\{x_j\}$ and therefore the $\{\xi_j\}$ are distinct, and $c_j \neq 0$ for $j = 1,\dots,n$. This implies that $\mathrm{rank}(\mathbf{G}) = n{+}1$ and $\mathrm{rank}(\mathbf{Y}) = n{+}1$ if $c_0 \neq 0$, otherwise $\mathrm{rank}(\mathbf{Y}) = n$. Therefore, $n \leq p \equiv \mathrm{rank}(\mathbf{H}) \leq n{+}1$ reflects the number of contributions in (1.1b).

We introduce the *singular value decomposition (SVD)*,

$$\mathbf{H} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top \tag{2.6a}$$

where $\mathbf{U} \in \mathbb{R}^{q\times(K+1)}$ and $\mathbf{V} \in \mathbb{R}^{(K+1)\times(K+1)}$ have orthonormal columns, and $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1,\dots,\sigma_{K+1})$ with

$$\sigma_1 \geq \cdots \geq \sigma_p > 0, \quad \sigma_{p+1} = \cdots = \sigma_{K+1} = 0 \ . \tag{2.6b}$$

Now consider "real" data with errors, (1.1a). The white noise assumption implies that we can expect the noise to be evenly distributed over all SVD-components. Assuming that $n \leq 5$, i.e. $\mathrm{rank}(\overline{\mathbf{H}}) \leq 6$, where $\overline{\mathbf{H}}$ is the unperturbed matrix, we introduce the threshold[1)]

$$\tau = 1.5\sigma_7 \ \ (= \max\{\sigma_7,\dots,\sigma_{K+1}\}) \tag{2.7a}$$

as a measure of the noise level. With respect to this, the *effective rank* $\widetilde{p}$ of $\mathbf{H}$ is defined by

$$\sigma_1 \geq \cdots \geq \sigma_{\widetilde{p}} > \tau \ . \tag{2.7b}$$

In the remainder we shall use p for $\widetilde{p}$.

Now, $\overline{\mathbf{H}}$ is approximated by the *truncated SVD expansion*

$$\widetilde{\mathbf{H}} = \widetilde{\mathbf{U}}\widetilde{\boldsymbol{\Sigma}}\widetilde{\mathbf{V}}^\top = \sum_{j=1}^{p} \sigma_j \mathbf{U}_{:,j}\mathbf{V}_{:,j}^\top \ . \tag{2.7c}$$

This approximation is inserted in (1.3),

$$\begin{aligned} &\widetilde{\mathbf{U}}\widetilde{\boldsymbol{\Sigma}}\mathbf{V}_{2:K+1,1:p}^\top\mathbf{b} = \widetilde{\mathbf{U}}\widetilde{\boldsymbol{\Sigma}}\mathbf{V}_{1,1:p}^\top \\ \Downarrow& \\ &\mathbf{V}_{2:K+1,1:p}^\top\mathbf{b} = \mathbf{V}_{1,1:p}^\top \ . \end{aligned} \tag{2.8a}$$

The derivation follows from the fact that $\widetilde{\mathbf{U}}^\top\widetilde{\mathbf{U}} = \mathbf{I}$ and $\widetilde{\boldsymbol{\Sigma}}$ is nonsingular. A similar derivation for the forward predictor shows that

1) Note that the number 7 and factor 1.5 in (2.7a) could be changed in other applications, cf. p. 18.

$$\mathbf{V}_{1:K,1:p}^{\mathsf{T}}\mathbf{f} = \mathbf{V}_{K+1,1:p}^{\mathsf{T}} \ . \tag{2.8b}$$

These two systems are underdetermined (p equations in K unknowns), and we use the *minimum norm solution* for $\mathbf{b}$ and $\mathbf{f}$, respectively. Rather than giving a thorough discussion (see e.g. [3] for that) we shall consider an example that displays some typical features.

3. Example

Let

$$M(\mathbf{x}, \mathbf{c}, t_i) = 10^{-2} + 2e^{-0.5t_i} + 4e^{-t_i} + 8e^{-2t_i} \ , \tag{3.1a}$$

with $t_i = 0.01i$. The $\{y_i\}$ are found by rounding the M-values to 6 decimal digits and adding further noise, so that

$$y_i = M(\mathbf{x}, \mathbf{c}, t_i) + \varepsilon_i^{(\mathrm{r6})} + \varepsilon\nu_i \ , \tag{3.1b}$$

where the $\{\nu_i\}$ are normal random numbers with mean zero, generated by MATLABs `randn` and normalized so that $\|\boldsymbol{\nu}\|_2 = 1$. Figure 3.1 shows the data for $\varepsilon = 10^{-3}$.

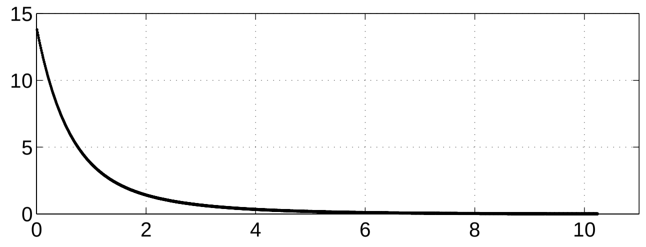

Figure 3.1. *Data for* $\varepsilon = 10^{-3}$

In Figure 3.2 we give the singular values for fixed K and three values of ε. Except for the first few singular values the $\{\sigma_j\}$ are almost constant. This reflects the "background noise". For $\varepsilon \le 10^{-6}$ the noise level is about 10^{-6} (caused by the rounding to 6 digits)

and we recognize that $\mathbf{H}$ has effective rank $p = 4$, which is equal to the number of parameters in M above. For $\varepsilon = 10^{-3}$ and $\varepsilon = 10^{-1}$ the background noise is of the same order of magnitude as ε, the dominant singular values are almost unchanged, but the smaller values "drown" in the background noise, so that the effective rank reduces to 3 and 2, respectively.

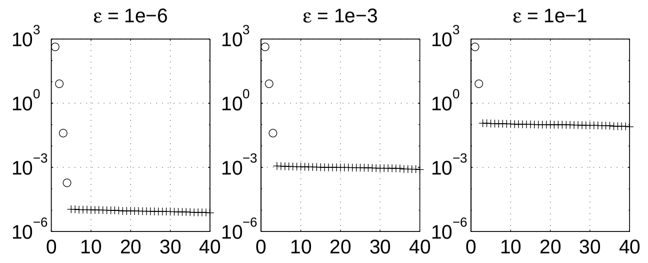

Figure 3.2. *Singular values.* $K = 40$

In the remainder of this section we look at the problem for $\varepsilon = 10^{-3}$. By increasing K it is possible to get information about the true number of terms in M: Figure 3.3 illustrates the effective rank of $\mathbf{H}$ as K grows. For some K-value between 40 and 80 the 4th contribution to M appears, and for larger K values it distinguishes better from the background noise. This is typical: with a larger "window" the effect of the noise in the data is decreased.

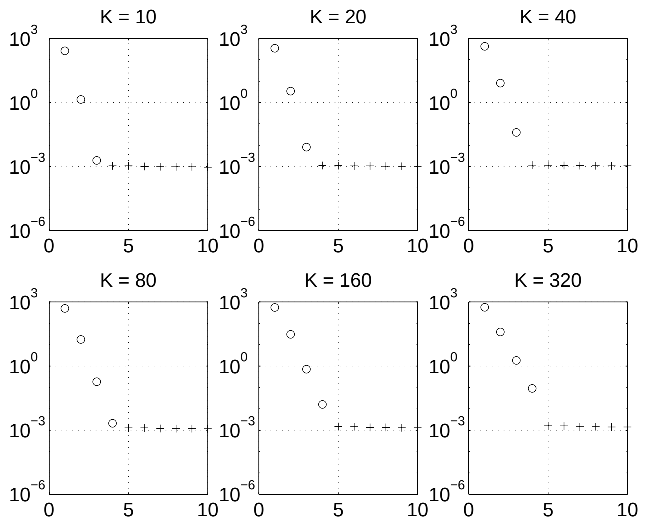

Figure 3.3. *Singular values $\sigma_1, \ldots, \sigma_{10}$ for increasing K*

The behaviour is further illustrated in Figures 3.4–5, where we give the first 6 columns in $\mathbf{U}$ and $\mathbf{V}$ for $K = 40, 80, 160$. Notice how $\mathbf{u}_1 = \mathbf{U}_{:,1}$ reflects the shape of the data in Figure 3.1 and how the fourth vectors are "purified" as we increase K. For all the K-values vectors nos. 5 and 6 look like noise.

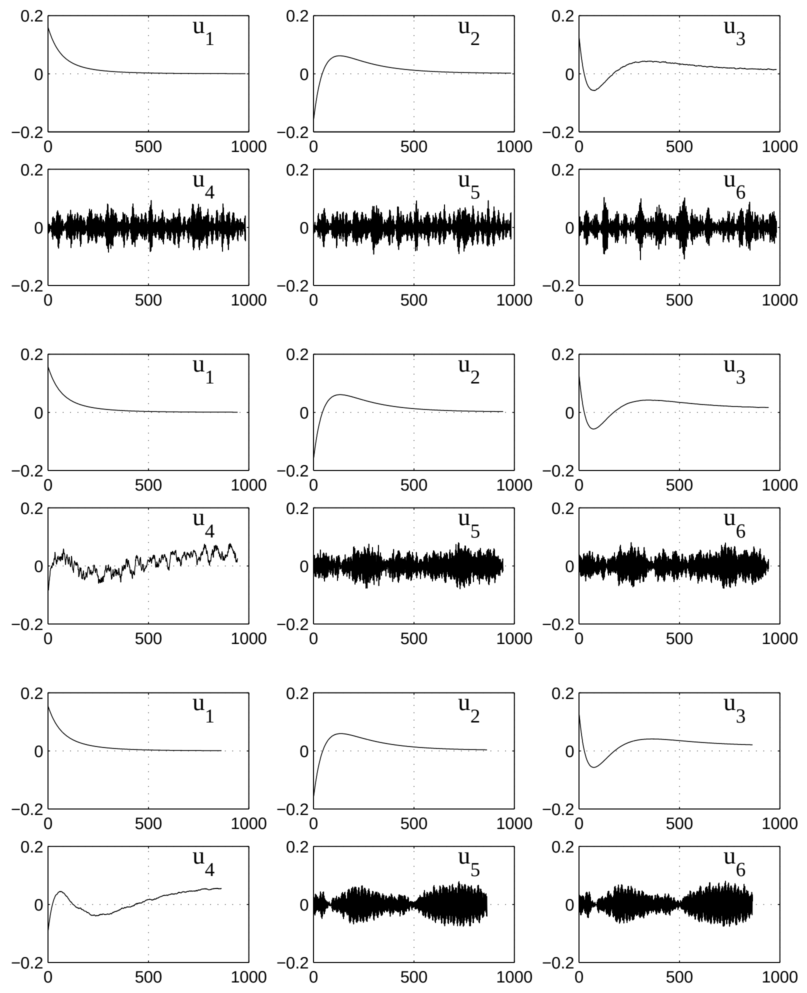

Figure 3.4. *Left singular vectors. $K = 40$ (top), $K = 80$ (middle) and $K = 160$ (bottom)*

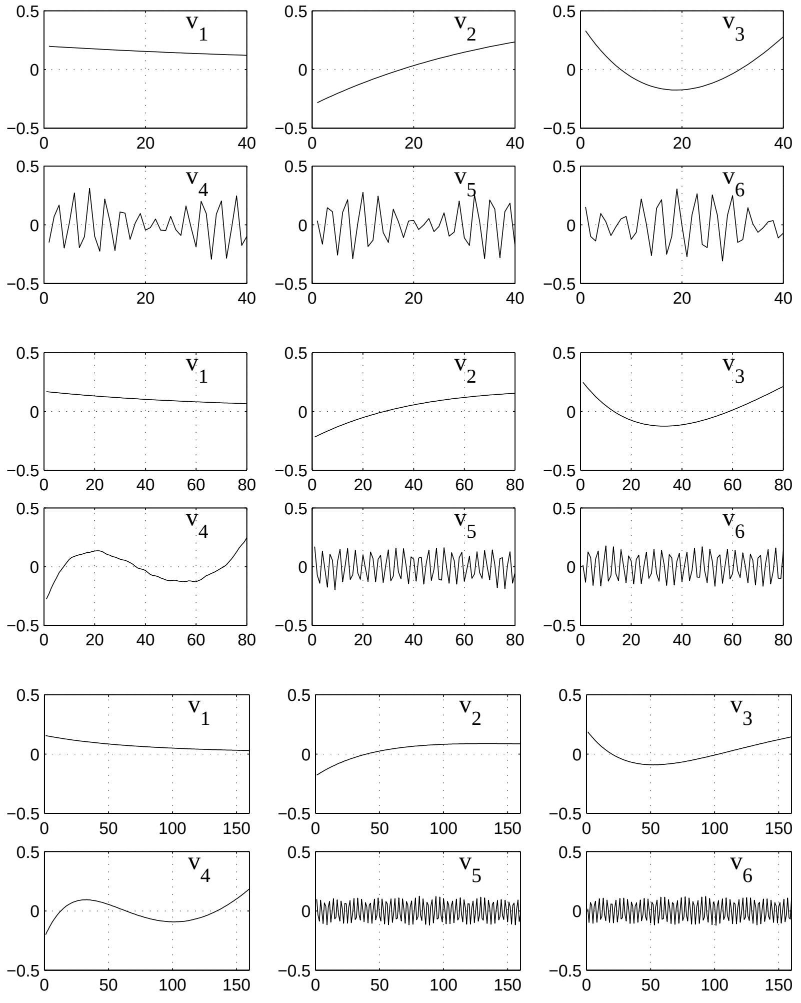

Figure 3.5. *Right singular vectors.* $K = 40$ *(top),* $K = 80$ *(middle) and* $K = 160$ *(bottom)*

Next, in Figure 3.6 we show the roots of (2.2b) and (2.3b) with the coefficients computed by (2.8) in the case $K = 40$.

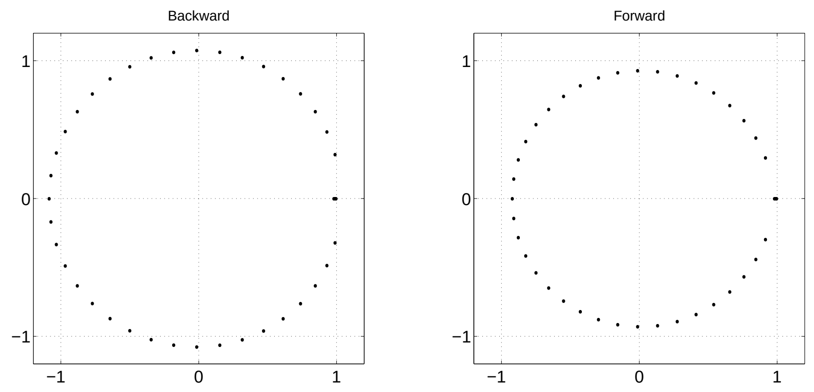

Figure 3.6. *Roots of* $\widetilde{P}_{40}$ *and* P_{40}

All roots except for the interesting ones lie close to the unit circle, respectively outside and inside for the backward and forward predictor. They are located between two circles centered in Origo and with radii $\underline{r}$ and $\overline{r}$ that tend to 1 as K grows:

	$K=10$	$K=20$	$K=40$	$K=80$	$K=160$	$K=320$
	Backward predictor					
$\underline{r}$	1.2442	1.0809	1.0396	1.0244	1.0095	1.0050
$\overline{r}$	1.3036	1.1392	1.0851	1.0537	1.0279	1.0163
	Forward predictor					
$\underline{r}$	.83247	.87919	.92164	.95429	.97290	.98400
$\overline{r}$	.86655	.92652	.96197	.98164	.99057	.99498

Table 3.1. *Inner and outer radius*

The interesting roots are real and in the range $0 < \xi_j \le 1$. For these we find the exponents as $x_j = (\ln \xi_j)/h$. We get the following results

j	$K=10$	$K=20$	$K=40$	$K=80$	$K=160$	$K=320$
			Backward predictor			
0				⇔0.2003	⇔0.0080	⇔0.0026
1	⇔0.6228	⇔0.4152	⇔0.4244	⇔0.5671	⇔0.5026	⇔0.5009
2	⇔1.7818	⇔0.9069	⇔0.9173	⇔1.0638	⇔1.0044	⇔1.0019
3	⇔2.4954	⇔1.9897	⇔1.9912	⇔2.0053	⇔2.0009	⇔2.0005
			Forward predictor			
0				⇔0.0394	⇔0.0007	⇔0.0021
1	+0.9617	⇔0.4319	⇔0.4154	⇔0.5261	⇔0.5014	⇔0.5008
2	⇔0.6954	⇔0.9325	⇔0.9027	⇔1.0435	⇔1.0034	⇔1.0018
3	⇔1.9469	⇔1.9952	⇔1.9875	⇔2.0061	⇔2.0008	⇔2.0005

Table 3.2. *Estimated exponents* $\{x_j\}$

Comparing with the values in (3.1) we see that already for $K = 40$ we have reasonably good approximations to the three exponents, while the constant term in (1.1b) has slower convergence. Also note that for the larger K-values the forward predictor seems to give slightly better approximations than the backward predictor.

The coefficients $\{c_j\}$ can be found by solving the least squares problem corresponding to (1.1). If we consider the $\{x_j\}$ as known (accept approximate values from Table 3.2), then we just have to solve a linear problem. Otherwise, as mentioned in the introduction, we can take these values as starting point for a nonlinear least squares solver, which is described in Section 5.

For illustration take the forward predictor values for $K = 320$. The linear least squares solution is

$$c^*_{0:3} = \left(1.0071{\cdot}10^{-2},\ 2.0104,\ 3.9969,\ 7.9927\right) .$$

The estimated standard deviation is $s = 3.44{\cdot}10^{-5}$, and in Figure 3.7 we give the residuals.

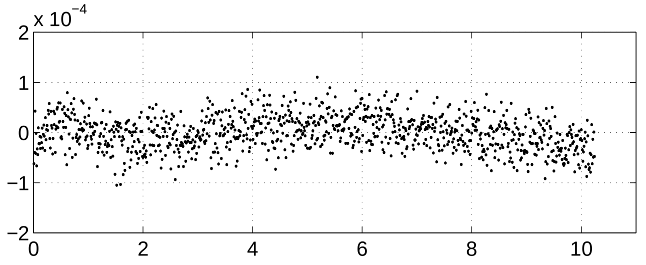

Figure 3.7. *Residuals. Linear fit, forward predictor,* $K = 320$

For the nonlinear approach we use the iterative method discussed in Section 5. Figure 3.8 shows residuals for a starting guess corresponding to $K = 80$ and for the solution.

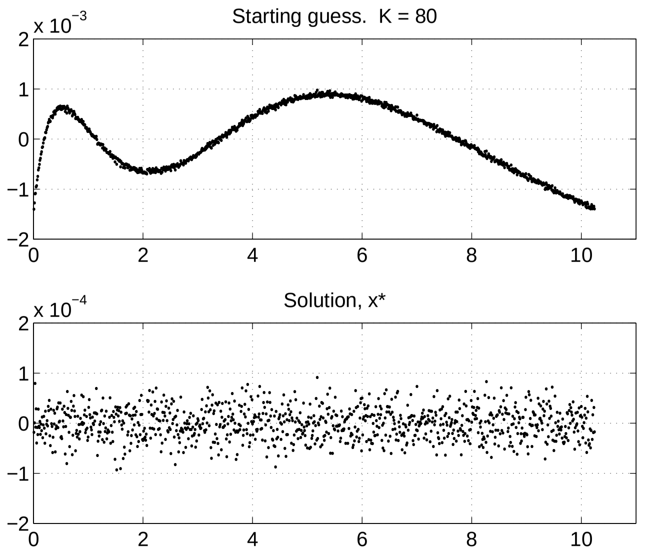

Figure 3.8. *Residuals. Nonlinear fit*

The solution has the estimated standard deviation $s = 3.12{\cdot}10^{-5}$ and the parameters

j	x_j^*	c_j^*
0	0	$1.0007 \cdot 10^{-2}$
1	−0.5002	2.0034
2	−1.0008	4.0011
3	−2.0003	7.9956

Table 3.3. *Nonlinear least squares fit*

For comparison, if we compute the linear fit with the "true" parameter values from (3.1a), we get $s = 3.13 \cdot 10^{-5}$, indicating that the rounding to 6 decimals and the further added noise has a (small) effect on the "best" values of the parameters.

4. SVD Algorithm

The discussion in the previous section shows that in order to get a reliable determination of the number of terms we have to take K large. Then the computation of the SVD decomposition of the matrix $\mathbf{H}$ (1.3b) is a considerable workload. Basically, the purpose is to find a set of p linearly independent vectors that span the same subspace as $\mathbf{H}$, with respect to the noise in the data. By taking into account the special features of the problem we can reach this goal in a cheaper way:

First, based on the experiences of Section 3 we concentrate on the forward predictor and look only at the first K columns in $\mathbf{H}$, i.e., we reformulate (1.3b–c) to

$$\mathbf{H}\mathbf{f} \simeq \begin{bmatrix} y_{1+K} \\ \vdots \\ y_{q+K} \end{bmatrix} \quad \text{with} \quad \mathbf{H} = \begin{bmatrix} y_1 & \cdots & y_K \\ \vdots & & \vdots \\ y_q & \cdots & y_{q+K-1} \end{bmatrix} . \tag{4.1}$$

Next, we still use the SVD decomposition if $K \le 10$, with (2.8) replaced by

$$\begin{aligned} &\widetilde{\mathbf{U}}\widetilde{\mathbf{\Sigma}}\widetilde{\mathbf{V}}^\mathsf{T}\mathbf{f} \simeq \mathbf{y}_{K+1:m} \\ \Downarrow \quad &\widetilde{\mathbf{V}}^\mathsf{T}\mathbf{f} = \widetilde{\mathbf{\Sigma}}^{-1}\widetilde{\mathbf{U}}^\mathsf{T}\mathbf{y}_{K+1:m} \equiv \boldsymbol{\eta} \\ \Downarrow \quad &\mathbf{f}_{\mathrm{mn}} = \widetilde{\mathbf{V}}\boldsymbol{\eta} \ , \end{aligned} \tag{4.2}$$

where $\mathbf{f}_{\mathrm{mn}}$ is the minimum norm solution.

For $K > 10$ we know that $p < 10$, and start by doing a series of 10 Householder reflections with column pivoting. We can express this in the form

$$\mathbf{Q}^\mathsf{T}\mathbf{H}\mathbf{P} = \begin{bmatrix} \mathbf{R} & \mathbf{W} \\ \mathbf{0} & \mathbf{E} \end{bmatrix} , \tag{4.3a}$$

where $\mathbf{Q}$ is orthogonal, $\mathbf{P}$ is a permutation matrix, $\mathbf{R} \in \mathbb{R}^{10\times 10}$ is upper triangular, and the elements of $\mathbf{E} \in \mathbb{R}^{(q-10)\times(K-10)}$ reflect the noise level. This means that the first 10 rows of the transformed matrix hold the significant information of $\mathbf{H}$. In particular, we can find the effective rank by computing the SVD of this submatrix,

$$\mathbf{A}^\mathsf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{W} \end{bmatrix}^\mathsf{T} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathsf{T} \ , \tag{4.3b}$$

and proceed as in (2.7), except that experiments showed that the factor 1.5 should be changed to 1.25. Then (4.2) is modified to

$$\begin{aligned} &\widetilde{\mathbf{V}}\widetilde{\mathbf{\Sigma}}\widetilde{\mathbf{U}}^\mathsf{T}\mathbf{P}^\mathsf{T}\mathbf{f} \simeq \left(\mathbf{Q}^\mathsf{T}\mathbf{y}_{K+1:m}\right)_{1:10} \equiv \boldsymbol{\zeta} \\ \Downarrow \quad &\mathbf{f}_{\mathrm{mn}} \simeq \mathbf{P}\widetilde{\mathbf{U}}\left(\widetilde{\mathbf{U}}^\mathsf{T}\widetilde{\mathbf{\Sigma}}^{-1}\widetilde{\mathbf{V}}^\mathsf{T}\boldsymbol{\zeta}\right) . \end{aligned} \tag{4.3c}$$

The vector $\boldsymbol{\zeta}$ can advantageously be computed together with the transformation (4.3a).

Another "costly" part of the algorithm is the computation of all roots of a polynomial of high order. We are only interested in the real roots close to 1, and we can easily find them by a Newton-Raphson iteration with deflation.

Finally, the number m of data points is typically very large. A simple reduction that keeps the basic properties of the problem is just to take every second data point. In terms of (2.1b) we aim for $\{\xi_j^2\}$, and for a given K the matrix $\mathbf{H}$ (4.1) "spans" twice the t-range. Therefore, we may expect a better smoothing effect, and this is verified in Section 6. Note, however, that if the data contains a small component that decays fast (i.e. c_j is relatively small and $\Leftrightarrow x_j$ is relatively large), then this strategy may lead to overlooking this component, see Section 7.

We have also tried proper smoothing of the data. Below we show a logarithmic plot of the data in Figure 3.1. It indicates that a piecewise linear approximation should be very accurate for exact data, and by proper choice of weights we can get a smoothing effect. We have experimented with simple expressions of the form

$$\begin{aligned} &\widehat{y}_i = e^{\eta_i}, \quad i=2,4,\ldots,m\Leftrightarrow 1 \\ &\text{with} \quad \eta_i = \tfrac{1}{w_0+2w_1}\big(w_0 \ln y_i + w_1(\ln y_{i-1} + \ln y_{i+1})\big) \ . \end{aligned} \tag{4.4}$$

Here, $\{\widehat{y}_i\}$ are the smoothed values. The selection in the previous paragraph also has this form with $w_0=1,\ w_1=0$.

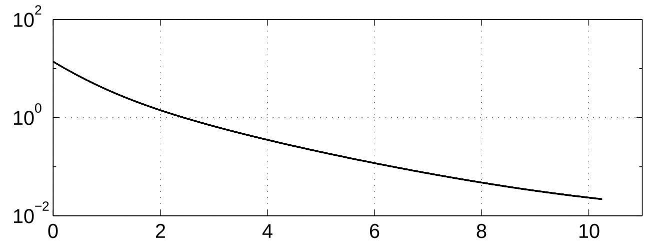

Figure 4.1. *Logarithmic plot of the data from Figure 3.1*

Experiments with this kind of smoothing was discouraging, see Section 6.

5. Least Squares Fit

Let

$$\mathbf{r} = \mathbf{r}(\mathbf{x},\mathbf{c}) \quad \text{with} \quad r_i = y_i \Leftrightarrow M(\mathbf{x},\mathbf{c},t_i),\ i=1,\ldots,m\ , \tag{5.1a}$$

where

$$M(\mathbf{x},\mathbf{c},t_i) = c_0 + \sum_{j=1}^{n} c_j e^{x_j t_i}\ ; \quad \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix},\quad \mathbf{c} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} . \tag{5.1b}$$

We want to determine the parameters $\mathbf{x}^*$, $\mathbf{c}^*$ that minimize

$$\phi(\mathbf{x},\mathbf{c}) \equiv \tfrac{1}{2}\mathbf{r}(\mathbf{x},\mathbf{c})^{\top}\mathbf{r}(\mathbf{x},\mathbf{c})\ . \tag{5.2}$$

This is a nonlinear problem, which may be solved in a number of ways. We exploit that the problem is *separable*: The coefficients c_j appear linearly, and as in [11] we let them be functions of $\mathbf{x}$ rather than letting them be part of the set of nonlinear parameters. For the current iterate $\mathbf{x}$ consider the linear problem

$$\mathbf{F}(\mathbf{x})\mathbf{c} \simeq \mathbf{y} \quad \text{with} \quad \mathbf{F}_{i,:} = \begin{bmatrix} 1 & e^{x_1 t_i} & \cdots & e^{x_n t_i} \end{bmatrix}\ . \tag{5.3}$$

We let $\mathbf{c}(\mathbf{x})$ be the least squares solution to this overdetermined system of equations. It can be computed via orthogonal transformation, cf. (4.3),

$$\mathbf{Q}^{\top}\mathbf{F}\mathbf{P} = \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix} \quad \Leftrightarrow \quad \mathbf{F} = \mathbf{Q}\begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix}\mathbf{P}^{\top}\ , \tag{5.4a}$$

where $\mathbf{R} \in \mathbb{R}^{(n+1)\times(n+1)}$ is upper triangular (and nonsingular). Then

$$\mathbf{c}(\mathbf{x}) = \mathbf{P}\mathbf{R}^{-1}\left(\mathbf{Q}^{\top}\mathbf{y}\right)_{1:n+1} \tag{5.4b}$$

and

$$\mathbf{r}(\mathbf{x},\mathbf{c}(\mathbf{x})) = \mathbf{r}(\mathbf{x}) = \mathbf{y} \Leftrightarrow \mathbf{F}(\mathbf{x})\mathbf{c}(\mathbf{x})\ . \tag{5.4c}$$

We also need the *Jacobian* $\mathbf{J}(\mathbf{x})$, containing the partial derivatives of the residuals with respect to the parameters. This is an $m\times n$ matrix, and by applying the ideas of [11, Sections 2 – 3] we find that

$$\mathbf{J} = \mathbf{G}\mathbf{C} \Leftrightarrow \mathbf{F}\mathbf{C}' \ , \tag{5.5a}$$

where

$$\mathbf{G} = \Leftrightarrow \mathrm{diag}(t_{1:m})\mathbf{F}_{:,2:n+1}, \quad \mathbf{C} = \mathrm{diag}(c_{1:n}) \ , \tag{5.5b}$$

and the matrix $\mathbf{C}' \in \mathbb{R}^{(n+1)\times n}$ solves the equation

$$\begin{aligned} &(\mathbf{F}^\top\mathbf{F})\mathbf{C}' = \mathbf{F}^\top\mathbf{G}\mathbf{C} \Leftrightarrow \mathrm{diag}(\mathbf{G}^\top\mathbf{r}) \equiv \mathbf{B} \\ \Updownarrow \quad & \\ &(\mathbf{R}^\top\mathbf{R})\,(\mathbf{P}^\top\mathbf{C}') = \mathbf{P}\mathbf{B} \ . \end{aligned} \tag{5.5c}$$

Thus, after having computed $\mathbf{B}$, the matrix $\mathbf{C}'$ is found by a forward and a back substitution followed by row interchanges. Note that this use of the factorization (5.4a) enhances accuracy and saves computational effort.

In contrast to the SVD analysis we do need equidistant t_i-values. Without significant effect on the results we can save some computational effort if we use close t_i-values where the model values vary most (for small t), while we can use larger spacing at the "tail". We shall illustrate this in Section 7.

The results in this report were found by using this formulation combined with a standard Levenberg-Marquardt-Gauss-Newton method [10], tuned to take into account that all components of $\mathbf{x}$ should be negative.[2)]

[2)] A Matlab implementation is available at
http://www.imm.dtu.dk/~hbn/Software/mexpfit.m
It needs the function implementing the QR-factorization
http://www.imm.dtu.dk/~hbn/Software/qrfacsol.m

6. Example. II

We have used the ideas from Section 4 on the data from Section 3. A plot of the the singular values computed by (4.3a-b) is indistinguishable from Figure 3.3, and in Table 4.1 we give the number of flops in Matlab needed to compute $\mathbf{f}$ by means of (4.3), compared with a full SVD of $\mathbf{H}$, (4.1), followed by (4.2). Also, we compare the number of flops needed to find all the roots of $\widetilde{P}$ by means of the Matlab function `roots`[3)] with the number of flops needed to find the p interesting roots by means of Newton-Raphson's method.

K	full SVD	Alg. (4.3)	K roots	p roots	p
10	1.08e6	1.08e6	2.13e4	1.33e3	3
20	4.57e6	9.06e5	1.33e5	2.62e3	3
40	1.93e7	1.95e6	8.44e5	5.26e3	3
80	7.38e7	3.91e6	6.12e6	1.10e4	3
160	2.86e8	7.31e6	4.39e7	2.96e4	4
320	1.01e9	1.21e7	3.20e8	6.51e4	4

Table 6.1. *Number of flops used to compute* $\mathbf{f}$ *and roots of* $\widetilde{P}_K$

Table 6.2 gives the estimated exponents. By comparison with the lower part of Table 3.2 we notice a good agreement.

j	$K=20$	$K=40$	$K=80$	$K=160$	$K=320$
0				⇔0.0256	⇔0.0022
1	⇔0.5760	⇔0.4215	⇔0.4103	⇔0.5075	⇔0.5008
2	⇔1.4352	⇔0.9098	⇔0.8928	⇔1.0101	⇔1.0019
3	⇔2.1207	⇔1.9886	⇔1.9841	⇔2.0016	⇔2.0005

Table 6.2 *New algorithm. Estimated exponents* $\{x_j\}$

Next, Table 6.3 shows the effect of reducing the number of data points. By comparison with Table 6.2 we see a considerable improvement for $K \le 160$, while the results for $K = 320$ are poorer. The re-

[3)] This function computes the roots of a polynomial as the eigenvalues of the associated companion matrix.

duced data set contains $m = 511$ points, and these results indicate that the condition $K \le \frac{1}{2}m$ should be taken seriously.

j	$K=10$	$K=20$	$K=40$	$K=80$	$K=160$	$K=320$
0				⇐0.0395	⇐0.0039	⇐0.0057
1	⇐0.6060	⇐0.4203	⇐0.4112	⇐0.5070	⇐0.5009	⇐0.4916
2	⇐1.7170	⇐0.9042	⇐0.8941	⇐1.0053	⇐1.0014	⇐1.0913
3	⇐2.4544	⇐1.9863	⇐1.9845	⇐2.0002	⇐2.0003	⇐1.8179

Table 6.3. *Exponents estimated from* $\mathbf{y}_{2:2:1022}$

Finally, the simple smoothing (4.4) is **not** a good idea. In Table 6.4 we give results for some choices of the weights $\mathbf{w} = [w_0\ w_1]$.

j	$\mathbf{w} = [1\ 1]$	$\mathbf{w} = [2\ 1]$	$\mathbf{w} = [4\ 1]$	$\mathbf{w} = [8\ 1]$
0	⇐0.5340	⇐0.3955	⇐0.2788	⇐0.1876
1	⇐1.0265	⇐0.8639	⇐0.7369	⇐0.6458
2	⇐1.9954	⇐1.7451	⇐1.5125	⇐1.3170
3	⇐3.0180	⇐2.6937	⇐2.4213	⇐2.2332

Table 6.4. *Exponents estimated from* $\widehat{\mathbf{y}}_{2:2:1022}$. $K = 80$

The quality of the results increases as the ratio w_0/w_1 grows. In the limit $w_0/w_1 \to \infty$ we get the results from Table 6.3.

We have experimented with a number of other smoothing algorithms, e.g. fitting $\{y_i\}$ or $\{\ln y_i\}$ with cubic splines as described in [9, Chapter 5]. None of these experiments were successful.

7. Applications

The data in Figure 1.1 are from an NMR scanning of meat, performed as described in [5]. It contains a total of $m = 3072$ points. Not all of them, however, contribute significant information.

In many (most?) practical applications, the first few data points may have exceptionally large errors, e.g. because the apparatus has not been properly tuned. To guard against that we omit the first two

data points. Also, Figure 1.1 shows a long flat tail. We decide to keep only those points, that are larger than 0.1% of the first one. More specifically, we take point numbers $3, \ldots, \widetilde{m}$, defined by

$$\begin{aligned} & y_i > 0.001 y_3 \quad \text{for } i = 3, \ldots, \widehat{m}\ , \\ & \widehat{m} := \max\{\widehat{m},\ \min\{m, 1002\}\}\ , \\ & \textbf{if } \widehat{m} > 0.8m \textbf{ then } \widetilde{m} := m \textbf{ else } \widetilde{m} := \widehat{m}\ . \end{aligned} \tag{7.1}$$

The result is $\widetilde{m} = 1558$, and in Table 7.1 we give the exponents computed by the algorithm of section 4. We give values corresponding to two sets of "active" points, given by the indices

$$\Omega_1 = 3\!:\!1\!:\!\widetilde{m}\ , \qquad \Omega_2 = 3\!:\!2\!:\!\widetilde{m}\ . \tag{7.2a}$$

The number of points is $\overline{m}_1 = 1556$ and $\overline{m}_2 = 778$, respectively, and we take $K = \min\{300, \frac{1}{2}\overline{m}\}$. The result is $n_1 = 3$, $n_2 = 4$ and

j	Ω_1	Ω_2
1	-5.9738e+00	-5.1834e+00
2	-1.9766e+01	-1.7859e+01
3	-4.3148e+01	-3.0483e+01
4		-1.3096e+02

Table 7.1. *Exponents estimated via SVD*

We take the second set of exponents as starting point for the nonlinear least squares solver. As mentioned in Section 5 we do not have to take all data points, and we have tried three different choices of "active" indices, viz. Ω_1, Ω_2 and

$$\Omega_3 = 3\!:\!1\!:\!52, 54\!:\!2\!:\!152, \ldots, 768\!:\!16\!:\!\widetilde{m}\ , \quad \overline{m}_3 = 259\ . \tag{7.2b}$$

The results are given in Table 7.2 and the corresponding residuals are shown in Figure 7.1.

There is no significant difference between the estimated standard deviations and the three values for x_1 and x_2 agree to (almost) one digits accuracy, while the accordance is increasingly poorer for x_3 and x_4. This illustrates that multiexponential fitting inherently is an ill-posed problem, see e.g. the discussion in [6, pp 275 – 279].

	Ω_1		Ω_2		Ω_3	
j	x_j	c_j	x_j	c_j	x_j	c_j
0	0	6.16e+01	0	5.61e+01	0	8.31e+01
1	-6.88e+00	4.39e+03	-6.75e+00	4.24e+03	-7.59e+00	5.39e+03
2	-2.00e+01	4.27e+04	-1.98e+01	4.19e+04	-2.07e+01	4.35e+04
3	-4.26e+01	3.24e+03	-3.87e+01	4.20e+03	-6.33e+01	1.54e+03
4	-6.87e+02	1.53e+03	-7.19e+02	1.71e+03	-8.57e+02	1.71e+03
s	25.9		25.2		26.9	

Table 7.2. *Estimated parameters and standard deviation*

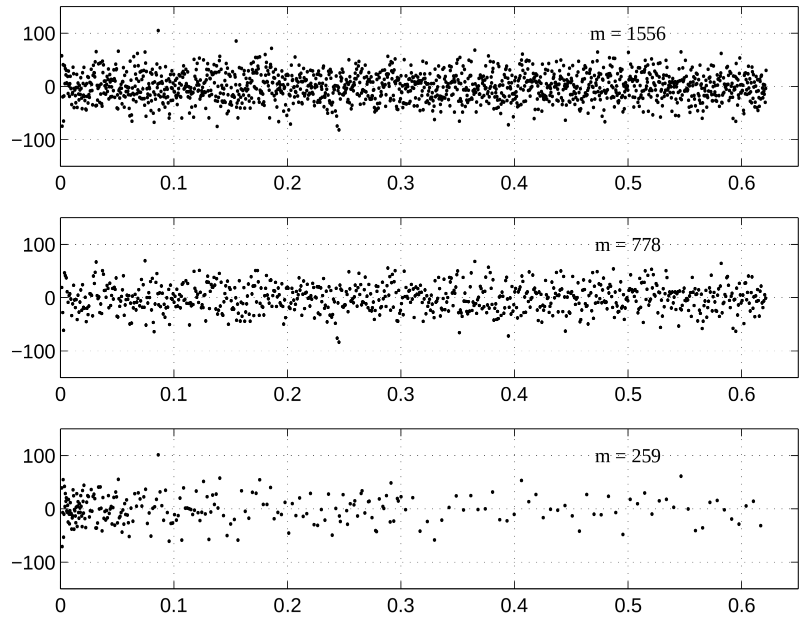

Figure 7.1. *Least squares residuals*

An inspection of Figure 7.1 shows that the full data set contains a number of *"wild points"* – points with abnormally large values for $|r_i|$, but here the positive and negative values seem to balance. For the Ω_2 set some of the large positive residuals have disappeared (their indices are not in Ω_2), and this is even more pronounced in the Ω_3 case. This lack of balance between positive and negative wild point residuals will pull the least squares fit downwards.

Further work should investigate the use of *robust estimation*, e.g. on the lines of [1], [7], [8], [4]. Until such an algorithm has been developed we suggest to use the choice Ω_2 as a reasonable compromise between robustness and computational efficiency.

Figure 7.2 illustrates that the tail is well modelled even though it did not take part in the fitting process. The first two data points were also ignored, and the absolute values of those residuals are considerably larger than the rest. It should be mentioned, however, that they are smaller than 0.5% of the corresponding ordinates.

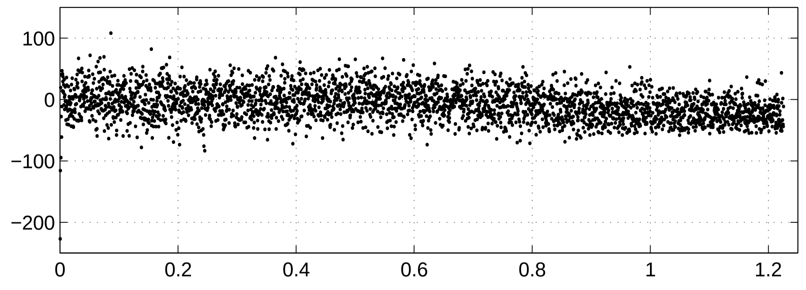

Figure 7.2. *Residuals for all data*

Figure 7.3 is a semilogarithmic plot of the "active" data and the five components of the model, as given by the Ω_2-values for **x** and **c** in Table 7.2. This illustrates why especially (x_4, c_4) are hard to find: Already from the start it is the smallest component, and it "dies out" very fast.

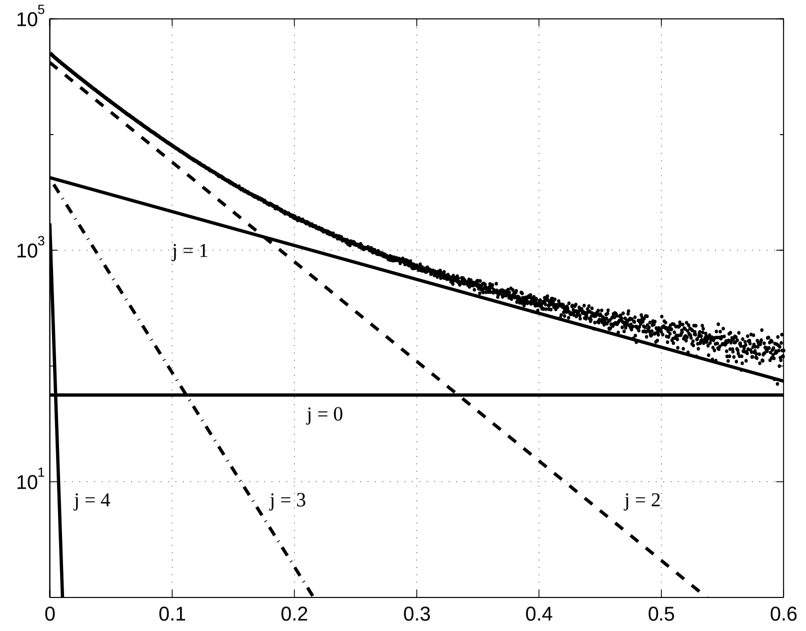

Figure 7.3. *Active data and components of fit*

Now, let us turn to the poor estimate in Table 7.1 obtained with the Ω_1 index set. Using this as starting point for the nonlinear least squares estimator with index set Ω_2 we get

x	0	`-8.45e+00`	`-2.16e+01`	`-2.99e+02`
c	`9.79e+01`	`7.07e+03`	`4.29e+04`	`1.33e+03`

(7.3a)

The estimated standard deviation is $s = 28.0$ and the residuals are shown below. They indicate a *trend* for $t \lesssim 0.3$, caused by overlooking one (or more) components.

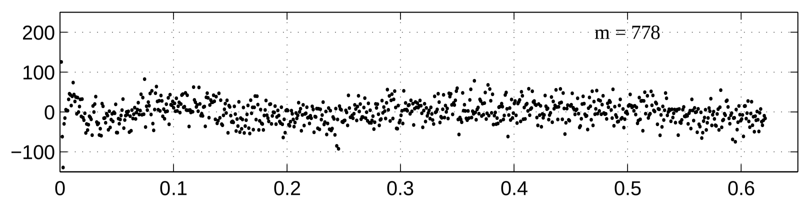

Figure 7.4. *Residuals for fit given by (7.3a)*

By applying SVD analysis to the residuals the trend can be quantified:

$$n_r = 2, \quad \mathbf{x}_r = \left[\texttt{-7.5899e+01} \quad \texttt{-8.3404e+02}\right]^{\mathsf{T}} . \tag{7.3b}$$

By comparing Table 2 and (7.3) we see that the latter has caught the slowest and the fastest component and has merged the two intermediate components into one. A new nonlinear least squares estimation with starting point given by the **x** of (7.3a) augmented by `-7.5899e+01` will give the solution from Table 7.2. An SVD analysis applied to the corresponding residuals gives $n_r = 0$ indicating a successful fitting.

We have tried the same approach on other "real life" problems and got similar results. We got all the data from Henrik T. Pedersen and Søren B. Engelsen, Food Technology, The Royal Veterinary and Agricultural University of Denmark. The values were computed with the index set Ω_1 if $\widetilde{m} < 1000$, otherwise Ω_2.

name	y_1	m	$\widetilde{m}$	n	n_r	s
filter	`6.87e+04`	3072	902	3	0	73.2
fish	`1.31e+04`	1024	1024	3	1	8.95
meat	`5.10e+04`	3072	1558	4	0	25.2
seed	`1.00e+04`	4096	3148	4	0	10.0
sugar	`6.78e+03`	512	512	2	0	4.33

Table 7.3. *Performance of algorithm*

8. Conclusion

Although we only have a limited experience with practical problems, the algorithm looks very promising and seems to be robust. Some parts are still under development. Among these are to find a better way of choosing which initial points to leave out (rather than just skipping the first two points). Also, it might be worthwhile to use a robust method instead of least squares estimation.

References

[1] H. Ekblom and K. Madsen (1989): *Algorithms for Non-Linear Huber Estimation.* BIT **29**, 60 – 76.

[2] H. Gesmar and P.C. Hansen (1994): *"Fast" Linear Prediction and its Application to NMR Spectroscopy.* J. MAGNETIC RESONANCE, Series A, **106**, 236 – 240.

[3] P.C. Hansen (1988): *Rank-Deficient and Discrete Ill-Posed Problems.* SIAM, Philadelphia.

[4] D. Hermey and G.A. Watson (1999): *Fitting Data with Errors in all Variables Using the Huber M-Estimator.* SIAM J. SCI. COMP. **20-4**, 1276 – 1298.

[5] S.M. Jepsen, H.T. Pedersen and S.B. Engelsen (1999): *Application of Chemometrics to Low-Field* ^1H *NMR Relaxation Data of Intact Fish Flesh.* J. SCI. FOOD AGRIC. **79**, 1793 – 1802.

[6] C. Lanczos (1957): *Applied Analysis.* Prentice-Hall, Englewood Cliffs, N.J.

[7] G. Li and K. Madsen (1988); *Robust Nonlinear Data Fitting.* In D.F. Griffiths and G.A. Watson (eds): *Numerical Analysis 1987*, Longman Scientific and Technical, London.

[8] K. Madsen and H.B. Nielsen (1990): *Finite Algorithms for Robust Linear Regression.* BIT **30**, 682 – 699.

[9] H.B. Nielsen (1998): *Cubic Splines.* Lecture notes, 2nd ed. IMM, DTU. 1 – 80.

[10] H.B. Nielsen (1999): *Damping Parameter in Marquardt's Method.* IMM, DTU. Report IMM-REP-1999-05. Available at http://www.imm.dtu.dk/~hbn/publ/TR9905.ps.Z

[11] H.B. Nielsen (2000): *Separable Nonlinear Least Squares.* IMM, DTU. Report IMM-REP-2000-01. Available at http://www.imm.dtu.dk/~hbn/publ/TR0001.ps.Z

[12] R. Prony (1795): *Essai Expérimental et Analytique sur les Lois de la Dilatibilité et sur celles de la Force Expansive de la Vapeur de l'Eau et de la Vapeur de l'Alkool, a Différentes Températures.* J. DE L'ÉCOLE POLYTECHNIQUE **1**, 24 – 76.