Notes on the square-root variable metric (SRVM) method for optimization
Carl Tape
August 16, 2007

This file: /home/carltape/latex/notes/tromp/square_root_variable_metric.tex
Matlab code: /opt/seismo-util/source/optim_matlab/optimization.m (plus additional functions)

Contents

1 Transcription of Hull and Tapley (1977) 2
1.1 Introduction .......................................................... 2
1.2 Square-Root Update .............................................. 4
1.3 Triangular Square-Root Update ................................. 6
1.4 Modified Square-Root Update ................................. 7
1.5 Conclusions ....................................................... 7

2 Comments on Hull and Tapley (1977) 8
2.1 Notation I ............................................................ 8
2.2 Notation II .......................................................... 9
2.3 Square-root variable metric (SRVM) algorithm (matrix version) .............. 10

3 Square-root variable metric (vector version) 12
3.1 Expanding the terms in the SRVM algorithm (matrix version) .................. 15
3.2 Expanding the terms in the SRVM algorithm (vector version) .................. 16
3.3 Sampling the posterior covariance matrix ................................ 17

4 Correspondence with Albert Tarantola 18

References 23

Note

In Section 1, I write out sections of Hull and Tapley (1977) using the notation of Tarantola (2005, Section 6.22). Some of the phrasing has been slightly altered for clarity (I hope!). In Section 2, I add some comments and additional derivations.

The papers of Hull and Tapley (1977) and Williamson (1975) have been cited (as of 2007) a total of six times in the literature. The paper of Morf and Kailath (1975) has been cited more than 75 times and may be a better source.
1 Transcription of *Hull and Tapley* (1977)

Square-root variable-metric methods for minimization

D. G. Hull and B. D. Tapley


Abstract

Variable-metric methods are presented which do not need an accurate one-dimensional search and eliminate roundoff error problems which can occur in updating the metric for large-dimension systems. The methods are based on updating the square-root of the metric, so that a positive-definite metric always results. The disadvantage of intentionally relaxing the accuracy of the one-dimensional search is that the number of iterations (and hence, gradient evaluations) increases. For problems involving a large number of variables, the square-root method is presented in a triangular form to reduce the amount of computation. Also, for usual optimization problems, the square-root procedure can be carried out entirely in terms of the metric, eliminating storage and computer time associated with computations of the square root of the metric.

1.1 Introduction

Over the past decade, considerable attention has been directed at the problem of minimizing nonlinear systems. When the function being minimized is highly nonlinear or ill-conditioned, the standard gradient or steepest-descent methods yield poor convergence characteristics. The Davidon–Fletcher–Powell (DFP) method (*Fletcher and Powell*, 1963) is the most widely adopted method for solving such problems and is acknowledged to have the best convergence characteristics.

The general problem which is considered can be described as follows:

Determine the value of the $M$-vector $m$ such that the scalar function value $S(m)$ is minimized.

In the subsequent discussion, it is assumed only that the function $S$ and the gradient $\hat{\gamma} = \partial S / \partial m$ exist for all $m$ in the range of interest. The standard DFP method for solving this problem is described by the following algorithm.
Begin with model \( m_0 \) and an initial positive-definite, symmetric matrix \( F_0 \) (usually \( F_0 = I \), where \( I \) is the identity matrix)\(^1\).

1. Compute the function value, \( S_k = S(m_k) \) and gradient \( \hat{\gamma}_k \), where
\[
\hat{\gamma}_k = \frac{\partial S}{\partial m}(m_k) .
\]
(1)

2. Compute the displacement \( \delta m_k \) from the formula
\[
\delta m_k = -\mu_k \hat{F}_k \hat{\gamma}_k = -\mu_k \phi_k ,
\]
where we have defined
\[
\phi_k = \hat{F}_k \hat{\gamma}_k .
\]
(3)

Use a one-dimensional search to determine the scalar stepsize \( \mu_k \) which minimizes \( S \) in the \( -\phi_k \) direction. Equation (2) can also be written in terms of the new model as
\[
m_{k+1} = m_k + \delta m_k = m_k - \mu_k \phi_k .
\]
(4)

3. Compute the new gradient,
\[
\hat{\gamma}_k = \frac{\partial S}{\partial m}(m_k) ,
\]
and form the difference
\[
\delta \hat{\gamma}_k = \hat{\gamma}_{k+1} - \hat{\gamma}_k .
\]
(6)

4. Compute a new metric \( \hat{F}_{k+1} \) from the relation\(^2\)
\[
\hat{F}_{k+1} = \hat{F}_k + \frac{\delta m_k \delta m_k^t}{\delta \gamma_k} - \frac{v_k v_k^t}{v_k^t \delta \gamma_k} ,
\]
(7)
where we have defined
\[
v_k = \hat{F}_k \delta \gamma_k .
\]
(8)

5. If the problem has not converged, repeat the procedure.

---

\(^1\)See note in Section 2.1.

\(^2\)Equation (7) can be found in Tarantola (2005) in three forms: general version, Eq. (6.343); hat-notation, Eq. (6.350); non-hat-notation, Eq. (6.357).
Two problems can arise with this algorithm. First, if the one-dimensional search is not conducted very accurately, the $\hat{F}$-matrix can become nonpositive-definite. Second, if the problem involves a large number of variables and if $\hat{F}$ is nearly singular at some point, roundoff errors in the computation of Equation (7) can cause $\hat{F}$ to become nonpositive-definite. In either case, a decrease in the function $S$ is no longer guaranteed, and the iteration process must be restarted with $\hat{F} = I$.

To allow a reduction in the accuracy with which the one-dimensional search must be made, Williamson (1975) has observed that the update formula (7) can be expressed in a square-root form, so that $\hat{F}_k$ will always be positive-definite. This procedure, which also eliminated the roundoff problems for large systems, is based on an equivalent form of Equation (7), that is, the well-known rank-one formula (see, for example, Davidon, 1968):

$$\hat{F}_{k+1} = \hat{F}_k - \frac{\hat{F}_k \hat{y}_k \hat{y}_k^t \hat{F}_k}{a_k} = \hat{F}_k - \frac{u_k u_k^t}{a_k},$$ (9)

where

$$\begin{align*}
\hat{y}_k &= \mu_k \hat{\gamma}_k + \delta \hat{\gamma}_k = \hat{\gamma}_k (\mu_k - 1) + \hat{\gamma}_{k+1} \quad (10) \\
v_k &= \hat{F}_k \delta \hat{\gamma}_k \quad (11) \\
u_k &= -\hat{F}_k \hat{y}_k \quad (12) \\
\quad = -\hat{F}_k (\mu_k \hat{\gamma}_k + \delta \hat{\gamma}_k) = -\mu_k \hat{F}_k \hat{\gamma}_k - \hat{F}_k \delta \hat{\gamma}_k = \delta m_k - v_k \quad (13) \\
a_k &= \hat{y}_k^t v_k. \quad (14)
\end{align*}$$

The equivalence of the two formulas has been established by Huang (1970).

The purpose of this paper is (i) to explore the extent to which the accuracy of the one-dimensional search can be relaxed using the square-root update, (ii) to reduce the amount of computer time required per iteration for large systems by modifying the square-root update, and (iii) to improve the square-root formulation for problems where roundoff error is not a factor.

1.2 Square-Root Update

As discussed by Williamson (1975), the algorithm developed by Potter and Stern (1963) (see Battin, 1964, p. 338-340) for updating the covariance matrix can be used to ensure that $\hat{F}$ remains positive definite, regardless of the accuracy with which the one-dimensional search is conducted. This is achieved by writing

$$\hat{F}_k = \hat{S}_k \hat{S}_k^t,$$ (15)

$\hat{S}_k$ is the square-root of the Hessian matrix $\hat{S}$.

---

$^3$The definitions for $u_k$, $v_k$, and $\delta_k$ are those of Tarantola (2005, Eq. 6.340-6.341). Note that Hull and Tapley (1977) do not show this many formulas. Note that $\hat{y}_k$ can be written as a weighted sum of gradients:

$$\hat{y}_k = \mu_k \hat{\gamma}_k + \delta \hat{\gamma}_k = \hat{\gamma}_k (\mu_k - 1) + \hat{\gamma}_{k+1}$$
where $\hat{S}_k$ is an $M \times M$ matrix, and finding the update formula for $\hat{S}_k$. It should be noted that, regardless of the formula used for $\hat{S}_k$, the matrix $\hat{F}_k$ will be positive-definite and symmetric.

To determine $\hat{S}_k$, Equation (9) is rewritten as

$$\hat{F}_{k+1} = \hat{S}_{k+1} \hat{S}^t_{k+1} = \hat{S}_k \left[ I - \frac{1}{a_k} \hat{S}^t_k \hat{y}_k \hat{y}^t_k \hat{S}_k \right] \hat{S}^t_k .$$

(16)

A solution for the term in brackets is sought such that

$$I - \frac{1}{a_k} \hat{S}^t_k \hat{y}_k \hat{y}^t_k \hat{S}_k = A_k A^t_k ,$$

(17)

where the matrix $A_k$ is defined as

$$A_k = I - \frac{\nu_k}{a_k} \hat{S}^t_k \hat{y}_k \hat{y}^t_k \hat{S}_k .$$

(18)

The constant $\nu_k$ is to be determined such that the assumed relation holds. Carrying out the multiplication on the right-hand side of Equation (17) leads to the following expression for $\nu_k$:

$$\nu_k = \frac{1 \pm (1 - b_k/a_k)^{1/2}}{b_k/a_k} ,$$

(19)

where

$$b_k = \hat{y}^t_k \hat{S}_k \hat{y}^t_k \hat{S}_k \hat{y}_k .$$

(20)

Either sign can be used, but the negative sign is chosen, because it makes $\nu_k$ continuous in the region of $b_k/a_k = 0$. As a consequence, the update formula for $\hat{S}_k$ becomes

$$\hat{S}_{k+1} = \hat{S}_k \left[ I - \frac{\nu_k}{a_k} \hat{S}^t_k \hat{y}_k \hat{y}^t_k \hat{S}_k \right] ,$$

(21)

and $\hat{F}_{k+1}$ is obtained through the multiplication

$$\hat{F}_{k+1} = \hat{S}_{k+1} \hat{S}^t_{k+1} .$$

(22)

It is seen from Equation (19) that the existence of $\hat{S}_k$ depends on the value of the quantity $b_k/a_k$ (see note 4).

1. If $b/a < 1$, $\nu$ is real, and an $\hat{S}$ exists.

2. If $b/a = 1$, $\hat{S}$ exists, but it is singular.

3. If $b/a > 1$, $\nu$ is imaginary, so that there is no real $\hat{S}$-matrix.

When the third case happens, a real $\hat{S}$-matrix must be generated to maintain a positive-definite

4In the remaining section, we omit the $k$ subscript to avoid clutter.
\( \hat{F} \)-matrix. As \( b/a \) varies in the range
\[
-\infty \leq b/a \leq 1,
\]
\( \nu \) takes on the values
\[
0 \leq \nu \leq 1.
\]
Hence, as proposed by Williamson (1975), a reasonable approach for the case where \( b/a > 1 \) is to use \( \nu = 1 \) and the computed value of \( a \) in Equation (21) for the ratio \( \nu/a \). If \( b/a = 1 \), a value of \( \nu \) such that \( \nu < 1 \) should be chosen to prevent \( \hat{S} \) from being singular. An alternative approach is to set \( \hat{S} \) equal to the identity matrix. One test with Rosenbrock’s function showed that the latter procedure required 39 iterations to achieve convergence, whereas the \( \nu = 1 \) procedure needed only 24 iterations. The explanation for this is that the \( \hat{S} \)-matrix generated with \( \nu = 1 \) retains some information about the geometry of the function \( S \), whereas using \( \hat{S} = I \) is the same as reinitializing the problem.

### 1.3 Triangular Square-Root Update

For large systems, the computation time for the square-root update can be reduced by employing an update formula in which \( \hat{S} \) is defined by an analytic Cholesky decomposition \( XXX \). Hence, \( \hat{S} \) is lower triangular, that is, all elements above the diagonal are zero. The triangular matrix representation for \( \hat{S} \) leads to a reduction in the amount of computation time required to generate the \( \hat{F} \)-matrix.

The derivation begins with Equation (16) rewritten in the form
\[
\hat{S}_{k+1} \hat{S}_{k+1}^t = \hat{S}_k \left[ I - \frac{1}{a_k} \mathbf{w}_k \mathbf{w}_k^t \right] \hat{S}_k^t, \tag{23}
\]
where
\[
\mathbf{w}_k = \hat{S}_k^t \hat{y}_k. \tag{24}
\]
Next, it is assumed that the term within brackets can be written as the product of a lower-triangular matrix \( \hat{B} \) and its transpose, that is,
\[
I - \frac{1}{a_k} \mathbf{w}_k \mathbf{w}_k^t = \hat{B}_k \hat{B}_k^t. \tag{25}
\]
Finally, this expression is expanded in scalar form, so that the elements \( \hat{B}_{ij} \) of the \( \hat{B} \) matrix can be determined. If
\[
\beta_1 = a \tag{26}
\]
\[
\beta_i = a - \sum_{k=1}^{i-1} w_k^2 \quad i = 2 \rightarrow n + 1, \tag{27}
\]
the expressions for \( \hat{B}_{ij} \) are given by

\[
\hat{B}_{ij} = \left( \frac{\beta_{i+1}}{\beta_i} \right)^{1/2} \quad i = 1 \rightarrow n , \quad (28)
\]

\[
\hat{B}_{ij} = \frac{-w_i w_j}{\beta_j \left( \frac{\beta_{i+1}}{\beta_i} \right)^{1/2}} \quad i = 2 \rightarrow n , j = 1 \rightarrow i - 1 \quad (29)
\]

As long as the \( \beta \)'s all have the same sign (either positive or negative), a \( \hat{B} \)-matrix can be determined, and the \( \hat{F} \)-matrix will be positive-definite. On the other hand, if the \( \beta \)'s change sign, some of the elements of \( \hat{B} \) become imaginary, which will lead to a nonpositive-definite matrix \( \hat{F} \).

MORE TEXT IN THIS SECTION (IS IT HELPFUL?)

1.4 Modified Square-Root Update

1.5 Conclusions
2 Comments on *Hull and Tapley* (1977)

2.1 Notation I

*Hull and Tapley* (1977) do not distinguish between the hat-notation and the non-hat-notation, as advocated in *Tarantola* (2005, Section 6.22). In our notation, the relationship between $F$ and $\hat{F}$ is

$$
\begin{align*}
\hat{F} &= F \ C_M \\
F &= \hat{F} \ C_M^{-1}.
\end{align*}
$$

The Hessian and curvature operators for the least-squares problem are (*Tarantola*, 2005, Eqs. 6.287–6.288)

Hessian (symmetric) \hspace{1cm} \hat{H} \simeq C_M^{-1} + G^t C_D^{-1} G = C_M^{-1} H \\
curvature (self–adjoint) \hspace{1cm} H = I + C_M G^t C_D^{-1} G = C_M \hat{H}.

The variable metric algorithms are designed to achieve

$$
\begin{align*}
\hat{F} &\rightarrow \hat{H}^{-1} \\
F &\rightarrow H^{-1}.
\end{align*}
$$

Thus, a suitable choice of initial pre-conditioning matrices might be

$$
\begin{align*}
\hat{F}_0 &= C_M \\
F_0 &= I.
\end{align*}
$$
2.2 Notation II

We wrote the transcription of Hull and Tapley (1977) using our own notation. From here onwards, we adapt the following set of variable changes:

\[
\begin{align*}
\hat{g}_k &= \hat{\gamma}_k - \hat{\gamma}_{k+1} = -\delta \hat{\gamma}_k \\
A_k &= -a_k \\
B_k &= b_k \\
C_k &= -\nu_k
\end{align*}
\]

This means that the expression

\[
\nu_k = \frac{1 \pm (1 - b_k/a_k)^{1/2}}{b_k/a_k}
\]

now becomes

\[
C_k = \frac{1 \pm (1 + B_k/A_k)^{1/2}}{B_k/A_k}
\]

The expression

\[
I - \frac{\nu_k}{a_k} w_k w_k^t
\]

now becomes

\[
I - \frac{C_k}{A_k} w_k w_k^t.
\]
2.3 Square-root variable metric (SRVM) algorithm (matrix version)

Here we summarize the algorithm presented in Hull and Tapley (1977). The square-root update modifies the original DFP algorithm to the following.

Begin with model \( m_0 \) and an initial positive-definite, symmetric matrix \( F_0 \). Let us choose \( F_0 = I \), and thus, considering previous equations, we have

\[
\begin{align*}
\hat{F}_0 &= C_M \\
\hat{S}_0 &= \left( \hat{F}_0 \right)^{1/2} = (C_M)^{1/2}
\end{align*}
\]

1. Compute the function value, \( S_k = S(m_k) \) and gradient \( \hat{\gamma}_k \), where

\[
\hat{\gamma}_k = \frac{\partial S}{\partial m}(m_k).
\]

2. Compute the new model

\[
m_{k+1} = m_k + \delta m_k = m_k - \mu_k \phi_k,
\]

where the search direction is given by

\[
\phi_k = \hat{F}_k \hat{\gamma}_k.
\]

The scalar stepsize \( \mu_k \), which minimizes \( S \) in the \(-\phi_k\) direction, is determined using a one-dimensional search.

3. Compute the new gradient,

\[
\hat{\gamma}_{k+1} = \frac{\partial S}{\partial m}(m_{k+1}),
\]

and form the difference

\[
\hat{g}_k = \hat{\gamma}_k - \hat{\gamma}_{k+1} = -\delta \hat{\gamma}_k.
\]
4. Compute $\hat{F}_{k+1}$ using the square-root update procedure.

\[
\begin{align*}
\hat{y}_k &= \mu_k \hat{y}_k - \hat{g}_k \\
\hat{w}_k &= \hat{S}_k \hat{y}_k \\
A_k &= \hat{y}_k^t \hat{F}_k \hat{g}_k \\
B_k &= \hat{w}_k^t \hat{w}_k \\
C_k &= \frac{1 \pm (1 + B_k/A_k)^{1/2}}{B_k/A_k} \\
\hat{S}_{k+1} &= \hat{S}_k \left[ I - \frac{C_k}{A_k} \hat{w}_k \hat{w}_k^t \right] \\
\hat{F}_{k+1} &= \hat{S}_{k+1} \hat{S}_{k+1}^t = \hat{S}_k \left[ I - \frac{C_k}{A_k} \hat{w}_k \hat{w}_k^t \right] \left[ I - \frac{C_k}{A_k} \hat{w}_k \hat{w}_k^t \right] \hat{S}_k^t
\end{align*}
\]

5. If the problem has not converged, repeat the procedure. At convergence, after $K$ iterations, the matrix $\hat{F}_K$ is a good estimate for the posterior covariance, that is

\[ \hat{F}_K \simeq \hat{H}^{-1} = C_{\text{post}}. \]
3 Square-root variable metric (vector version)

For very large problems requiring tens of millions of parameters — for example, tomography in Southern California — it is not feasible to store the updating matrices $\hat{S}_k$, or to perform matrix operations with such large matrices. However, it is feasible to store a series of vectors and scalars that go into constructing it. Here we outline the procedure for what vectors must be stored.

We begin with model $m_0$, containing $M$ elements, and assume that the $M \times M$ covariance matrix $C_M$ is sparse — at least sparse enough to readily compute its square-root using a Cholesky decomposition. We choose an initial pre-conditioning matrix $F_0 = I$ (which is obviously symmetric positive-definite). Considering previous equations, we have

$$
\hat{F}_0 = C_M \\
\hat{S}_0 = \left(\hat{F}_0\right)^{1/2} = (C_M)^{1/2}.
$$

(35) (36)

For simplicity we will assume that the covariance matrix contains only the variances of the model parameters, that is:

$$
C_M = \begin{bmatrix}
\sigma_1^2 & 0 & 0 & \ldots & 0 \\
0 & \sigma_2^2 & 0 & \ldots & 0 \\
0 & 0 & \ddots & \vdots & \vdots \\
0 & 0 & \ldots & \sigma_{M-1}^2 & 0 \\
0 & 0 & \ldots & 0 & \sigma_M^2
\end{bmatrix}
$$

(37)

In this case,

$$
\hat{S}_0 = (C_M)^{1/2} = \text{diag}(\sigma),
$$

(38)

where “diag” denotes a diagonal matrix and $\sigma$ is defined as

$$\sigma = \begin{bmatrix}
\sigma_1 \\
\sigma_2 \\
\vdots \\
\sigma_M
\end{bmatrix},$$

such that, in index notation,

$$
\left(\hat{S}_0\right)_{ij} = \sigma_i \delta_{ij} \quad \text{(no summation)}.
$$

1. Compute the function value, $S_k = S(m_k)$ and gradient $\hat{\gamma}_k$, where

$$\hat{\gamma}_k = \frac{\partial S}{\partial m}(m_k).$$
2. Compute the new model
\[ \mathbf{m}_{k+1} = \mathbf{m}_k + \delta \mathbf{m}_k = \mathbf{m}_k - \mu_k \phi_k , \]
where the search direction is given by
\[ \phi_k = \hat{\mathbf{F}}_k \hat{\gamma}_k = \hat{\mathbf{S}}_k \hat{\mathbf{S}}_k^t \delta_k = \hat{\mathbf{S}}_k \delta_k , \]
where
\[ \delta_k = \hat{\mathbf{S}}_k \hat{\gamma}_k \, . \]

The scalar stepsize \( \mu_k \), which minimizes \( S \) in the \( -\phi_k \) direction, is determined using a one-dimensional search.

3. Compute the new gradient,
\[ \hat{\gamma}_{k+1} = \frac{\partial S}{\partial \mathbf{m}}(\mathbf{m}_{k+1}) \, , \] (39)
and form the difference
\[ \hat{\mathbf{g}}_k = \hat{\gamma}_k - \hat{\gamma}_{k+1} = -\delta \hat{\gamma}_k . \]

4. Compute \( \hat{\mathbf{S}}_{k+1} \) using the square-root update procedure.
\[
\begin{align*}
\hat{\mathbf{y}}_k &= \mu_k \hat{\gamma}_k - \hat{\mathbf{g}}_k \\
\mathbf{w}_k &= \hat{\mathbf{S}}_k \hat{\mathbf{y}}_k \\
\beta_k &= -\hat{\mathbf{S}}_k \hat{\mathbf{g}}_k = \mathbf{w}_k - \mu \hat{\mathbf{S}}_k \hat{\gamma}_k = \mathbf{w}_k - \mu \delta_k \\
A_k &= \hat{\mathbf{y}}_k^t \hat{\mathbf{S}}_k \hat{\mathbf{S}}_k^t \mathbf{g}_k = \mathbf{w}_k^t \beta_k \\
B_k &= \mathbf{w}_k^t \mathbf{w}_k \\
C_k &= 1 \pm (1 + B_k/A_k)^{1/2} \\
\hat{\mathbf{S}}_{k+1} &= \hat{\mathbf{S}}_k \left[ \mathbf{I} - \frac{C_k}{A_k} \mathbf{w}_k \mathbf{w}_k^t \right] .
\end{align*}
\]

5. If the problem has not converged, repeat the procedure. At convergence, after \( K \) iterations, the matrix \( \hat{\mathbf{F}}_K \) is a good estimate for the posterior covariance, that is
\[ \hat{\mathbf{F}}_K \simeq \hat{\mathbf{H}}^{-1} = \mathbf{C}_{\text{post}} . \]

The above algorithm does not require that \( \hat{\mathbf{S}}_k \) be stored in memory, but rather that the
following two operations be executed:

\[
\hat{S}_k(\cdot) \\
\hat{S}_k^t(\cdot),
\]

where (\cdot) denotes an arbitrary vector. When \(\hat{S}_k\) is expressed in terms of previously saved vectors and scalars \((w_k, A_k, C_k)\), as shown in Section 3.1, it becomes apparent that no matrix need be stored, and no matrix operation need be performed.
3.1 Expanding the terms in the SRVM algorithm (matrix version)

Here we show we may compute $\mathbf{S}_k$ from a set of stored vectors and scalars.

From previously we have

$$
\mathbf{S}_{k+1} = \mathbf{S}_k \begin{bmatrix} 
I & \frac{C_k}{A_k} \mathbf{w}_k \mathbf{w}_k^t 
\end{bmatrix}
$$

$$
\mathbf{w}_k = \mathbf{S}_k^t \hat{y}_k.
$$

(40)

The first update for $\mathbf{S}$ is given by

$$
\mathbf{S}_1 = \mathbf{S}_0 \begin{bmatrix} 
I & \frac{C_1}{A_1} \mathbf{w}_1 \mathbf{w}_1^t 
\end{bmatrix}
$$

$$
\mathbf{w}_1 = \mathbf{S}_0^t \hat{y}_1.
$$

(42)

The next update is given by

$$
\mathbf{S}_2 = \mathbf{S}_1 \begin{bmatrix} 
I & \frac{C_2}{A_2} \mathbf{w}_2 \mathbf{w}_2^t 
\end{bmatrix} = \mathbf{S}_0 \begin{bmatrix} 
I & \frac{C_1}{A_1} \mathbf{w}_1 \mathbf{w}_1^t 
\end{bmatrix} \begin{bmatrix} 
I & \frac{C_2}{A_2} \mathbf{w}_2 \mathbf{w}_2^t 
\end{bmatrix}
$$

$$
\mathbf{w}_2 = \mathbf{S}_1^t \hat{y}_2 = \left( \mathbf{S}_0 \begin{bmatrix} 
I & \frac{C_1}{A_1} \mathbf{w}_1 \mathbf{w}_1^t 
\end{bmatrix} \right)^t \hat{y}_2 = \begin{bmatrix} 
I & \frac{C_1}{A_1} \mathbf{w}_1 \mathbf{w}_1^t 
\end{bmatrix} \mathbf{S}_0^t \hat{y}_1.
$$

(44)

(45)

(46)

The next update is given by

$$
\mathbf{S}_3 = \mathbf{S}_2 \begin{bmatrix} 
I & \frac{C_3}{A_3} \mathbf{w}_3 \mathbf{w}_3^t 
\end{bmatrix} = \mathbf{S}_0 \begin{bmatrix} 
I & \frac{C_1}{A_1} \mathbf{w}_1 \mathbf{w}_1^t 
\end{bmatrix} \begin{bmatrix} 
I & \frac{C_2}{A_2} \mathbf{w}_2 \mathbf{w}_2^t 
\end{bmatrix} \begin{bmatrix} 
I & \frac{C_3}{A_3} \mathbf{w}_3 \mathbf{w}_3^t 
\end{bmatrix}
$$

$$
\mathbf{w}_3 = \mathbf{S}_2^t \hat{y}_3 = \left( \mathbf{S}_0 \begin{bmatrix} 
I & \frac{C_1}{A_1} \mathbf{w}_1 \mathbf{w}_1^t 
\end{bmatrix} \begin{bmatrix} 
I & \frac{C_2}{A_2} \mathbf{w}_2 \mathbf{w}_2^t 
\end{bmatrix} \right)^t \hat{y}_3
$$

$$
= \begin{bmatrix} 
I & \frac{C_2}{A_2} \mathbf{w}_2 \mathbf{w}_2^t 
\end{bmatrix} \begin{bmatrix} 
I & \frac{C_1}{A_1} \mathbf{w}_1 \mathbf{w}_1^t 
\end{bmatrix} \mathbf{S}_0^t \hat{y}_3.
$$

(47)

(48)

Thus, after $K$ iterations, the formulas are

$$
\mathbf{S}_K = \mathbf{S}_0 \begin{bmatrix} 
I & \frac{C_1}{A_1} \mathbf{w}_1 \mathbf{w}_1^t 
\end{bmatrix} \ldots \begin{bmatrix} 
I & \frac{C_k}{A_k} \mathbf{w}_k \mathbf{w}_k^t 
\end{bmatrix} \begin{bmatrix} 
I & \frac{C_K}{A_K} \mathbf{w}_K \mathbf{w}_K^t 
\end{bmatrix}
$$

$$
\mathbf{w}_K = \begin{bmatrix} 
I & \frac{C_{K-1}}{A_{K-1}} \mathbf{w}_{K-1} \mathbf{w}_{K-1}^t 
\end{bmatrix} \ldots \begin{bmatrix} 
I & \frac{C_k}{A_k} \mathbf{w}_k \mathbf{w}_k^t 
\end{bmatrix} \begin{bmatrix} 
I & \frac{C_1}{A_1} \mathbf{w}_1 \mathbf{w}_1^t 
\end{bmatrix} \mathbf{S}_0^t \hat{y}_K.
$$

(49)

(50)

Therefore, what is needed to reconstruct $\mathbf{S}_K$ is the following set of vectors and scalar values:

$$
\mathbf{w}_k \quad (k = 1, \ldots, K)
$$

$$
A_k, \ B_k, \ C_k \quad (k = 1, \ldots, K)
$$

Note that for $\mathbf{w}_K$, one needs to have stored $\hat{y}_K$. 
3.2 Expanding the terms in the SRVM algorithm (vector version)

Here we step through the SRVM algorithm using only vector operations and storing only vectors and scalars.

We begin with a diagonal \( \hat{S}_0 \), as defined by \( \sigma \) in Equation (38), as well as initial model \( m_0 \) and initial gradient \( \gamma_0 \). We compute the initial search direction, \( \phi_0 \), via

\[
\delta_0 = \hat{S}_0^t \gamma_0 \\
\phi_0 = \hat{S}_0 \delta_0 .
\]

We compute the new model via

\[
m_1 = m_0 - \mu_0 \phi_0 .
\]

(The scalar stepsize \( \mu_0 \), which minimizes \( S \) in the \( -\phi_0 \) direction, is determined using a one-dimensional search.) We then compute the new gradient

\[
\gamma_1 = \frac{\partial S}{\partial m}(m_1) ,
\]

and form the difference

\[
\delta_0 = \gamma_0 - \gamma_1 = -\delta \gamma_0 .
\]

We then compute the following:

\[
\hat{y}_0 = \mu_0 \gamma_0 - \delta_0 \\
w_0 = \hat{S}_0 \hat{y}_0 \\
\beta_0 = w_0 - \mu \delta_0 \\
A_0 = w_0' \beta_0 \\
B_0 = w_0' w_0 \\
C_0 = \frac{1 \pm (1 + B_0/A_0)^{1/2}}{B_0/A_0} ,
\]

and store \( w_0, A_0, \) and \( C_0 \).

\footnote{Keep in mind that \( \hat{S}_0 \) is chosen as a diagonal matrix, and thus a matrix operation (or storage) is not needed:}

\[
\phi_0 = \hat{S}_0 \delta_0 = \text{diag} (\sigma) \delta_0 \\
\delta_0 = \hat{S}_0^t \gamma_0 = \text{diag} (\sigma) \gamma_0 ,
\]

which, in index notation, is

\[
(\phi_0)_i = (\sigma)_i (\delta)_i \quad \text{(no summation),}
(\delta)_i = (\sigma)_i (\gamma_0)_i \quad \text{(no summation).}
\]
Next, update the search direction via
\[
\delta_1 = \mathbf{S}_1^t \hat{\gamma}_1 = \left[ \mathbf{I} - \frac{C_1}{A_1} \mathbf{w}_1 \mathbf{w}_1^t \right] \mathbf{S}_0^t \hat{\gamma}_1 = \left[ \mathbf{S}_0^t \hat{\gamma}_1 - \left( \frac{C_1}{A_1} \mathbf{w}_1 \mathbf{w}_1^t \right) \mathbf{w}_1 \right]
\]
\[
\phi_1 = \hat{\mathbf{S}}_1 \delta_1 = \hat{\mathbf{S}}_0 \left[ \mathbf{I} - \frac{C_1}{A_1} \mathbf{w}_1 \mathbf{w}_1^t \right] \delta_1 = \hat{\mathbf{S}}_0 \left[ \delta_1 - \left( \frac{C_1}{A_1} \mathbf{w}_1 \delta_1 \right) \mathbf{w}_1 \right].
\]

Update the model, gradient, and gradient difference:
\[
\mathbf{m}_2 = \mathbf{m}_1 - \mu_1 \phi_1 \\
\hat{\gamma}_2 = \frac{\partial \mathbf{S}}{\partial \mathbf{m}}(\mathbf{m}_2) \\
\hat{\mathbf{g}}_1 = \hat{\gamma}_1 - \hat{\gamma}_2 = -\delta \hat{\gamma}_1.
\]

We then compute the following:
\[
\hat{\mathbf{y}}_1 = \mu_1 \hat{\gamma}_1 - \hat{\mathbf{g}}_1 \\
\mathbf{w}_1 = \hat{\mathbf{S}}_1^t \hat{\mathbf{y}}_1 = \left[ \mathbf{I} - \frac{C_1}{A_1} \mathbf{w}_1 \mathbf{w}_1^t \right] \hat{\mathbf{S}}_0^t \hat{\mathbf{y}}_1 = \hat{\mathbf{S}}_0^t \hat{\mathbf{y}}_1 - \left( \frac{C_1}{A_1} \mathbf{w}_1 \hat{\mathbf{S}}_0^t \hat{\mathbf{y}}_1 \right) \mathbf{w}_1
\]
\[
\beta_1 = \mathbf{w}_1 - \mu \delta_1 \\
A_1 = \mathbf{w}_1^t \beta_1 \\
B_1 = \mathbf{w}_1 \mathbf{w}_1 \\
C_1 = \frac{1 \pm (1 + B_1/A_1)^{1/2}}{B_1/A_1},
\]
and store \(\mathbf{w}_1, A_1, \) and \(C_1.\)

In this manner, by storing the vectors \(\mathbf{w}_k\) and the scalars \(A_k\) and \(C_k,\) we can, in theory reconstruct the matrix \(\hat{\mathbf{S}}_k\) (and thus \(\hat{\mathbf{F}}_k\)) as shown in Equations (49) and (50). However, we are not interested in reconstructing this matrix \(\hat{\mathbf{F}}_k \approx \mathbf{C}_{\text{post}},\) which would be prohibitive in our problem, since \(\hat{\mathbf{S}}_k\) is \(M \times M,\) where \(M\) is on the order of \(10^8.\) We are interested in using \(\hat{\mathbf{S}}_k(\cdot)\) to generate random realizations of the model covariance, as discussed next.

### 3.3 Sampling the posterior covariance matrix

TEXT HERE.
4 Correspondence with Albert Tarantola

Email from AT 09-June-2007

Jeroen seems to be serious about not only producing a model that predicts seismograms close to the observed ones, but in also estimating uncertainties. And he accepts that the goal (reachable or not) has to be to produce a collection of samples of the posterior distribution in the model parameter space. Jeroen has chosen to be practical, and will start by solving an optimization problem, well far from any Monte Carlo method. For that, he has chosen a clever way to define a waveform fit (which is not directly fitting seismogram amplitudes), and has purchased a big computer. While I was at Caltech, I had numerous discussions with Jeroen and members of his team, and we ended by pointing our fingers to the square root variable metric optimization algorithm. While variable metric methods are well understood, the method consisting to update the square root of the metric is not. But this may be crucial: as we know, the output of the optimization algorithm is the mean of the posterior Gaussian, and if, besides that mean, we have the square root of the posterior covariance, it is very efficient to generate samples of the posterior distribution. One of Jeroen’s collaborators, Qinya Liu, is about to provide us with a numerical example of how this works. Two comments on this:

1. This is least-squares, and we know that least-squares only makes sense when the prior distribution in the model space is Gaussian. In our case, this amounts to say that we model the distribution of mass density by saying that its logarithm is a Gaussian random field, and the distribution of the stiffness tensor by saying that its logarithm is a Gaussian random field. This assumption is clearly untrue as soon as we are serious in considering the position of the main discontinuities (main faults, main reflectors) as basic model parameters. But those parameters are not very numerous, and I am quite confident that we will be able to explicitly deal with that.

2. In the context of least squares (i.e., forgetting the point just mentioned), there is one point that bothers me. For my lessons at Caltech, I developed a simple, one-dimensional, numerical example, taking as "true model" for the seismic velocity in the medium a random realization of a Gaussian random field with exponential covariance. This makes a very rough model (it is so rough that has no derivative at any of its points). Then, I integrate the wave equation to exactly model some seismograms, that, after the addition of some simulated noise, are to be used as the observations to solve the inverse problem. A least-squares optimization algorithm is implemented, and the best-fitting model is obtained. As this is a one-dimensional example, the posterior covariance can be explicitly evaluated, and its square root computed. With the output of the least-squares optimization algorithm (the mean of the posterior Gaussian field), and the square root of the posterior covariance function, it is possible to generate samples of the posterior Gaussian random field. When I do that, the models look right. But the residual seismograms produced by each of these models seem too large. In any case, larger than those that a plane Monte Carlo method would produce. From where my puzzling question.
**Question:** To produce the faked “observed seismograms” I have propagated waves in a very rough medium (a random sample of Gaussian random field with given mean function and given covariance function [the exponential one]). The random sample used is rough, while the mean is smooth. The optimization algorithm starts iterating at the prior mean (smooth) and converges to the posterior mean (also smooth). So in all the optimization process I am propagating waves in smooth media only, while I know the the ”observed seismograms” correspond to a rough medium. Should we face a linear relation between model parameters and observations, this would not be a problem. But my “real medium” is such that the Born approximation would not make any sense here: we are not inside the linear regime. Can we, in this situation, still say that if the prior distribution is Gaussian, the posterior distribution is approximately Gaussian?

I fear that the answer may be no. And if this is the answer, we would face a serious theoretical limitation to the use of the least-squares optimization techniques for waveform fitting (and for properly estimating uncertainties). Any thought on that, Klaus?

**Email from AT 11-June-2007**

I am afraid that to convince people to do something better that what they do today, we have to be very realistic in modeling the propagation of elastic waves. The first check could be to generate a synthetic test with a truly heterogeneous medium. I am a little bit skeptical about the possibility of properly solving then the waveform fitting problem if we only propagate waves in smooth media (as the least-squares method tells us to do...).

This, in particular, suggests that it could be dangerous that you limit your numerical tests to models that are as smooth as you think your solutions will be. Your models should be as heterogeneous as the real Earth can be: only then we will know how good can we be in interpreting seismological data.

**Email from Andre Journel to AT 13-June-2007**

Dear All,

Not being a geophysicist and being ignorant of the details of modeling wave equations, I hesitate to step in but do not wish Tarantola to think I am aloft.

A few disjoint remarks within my field of expertise:

1. A Gaussian random field with a zero nugget effect is derivable, actually infinitely so if its covariance is infinitely derivable as is the case for a Gaussian covariance model. So I do not believe that Albert’s random realization can qualify as being “rough”. Real heterogeneities, such as generated by faults, would call for non stationary, non Gaussian, models that cannot be defined with a mere covariance.

2. I doubt that the (non-linear) smoothing performed by the long waves would undo the non-linearity of small scale heterogeneities, in addition they would start seeing large scale heterogeneities which have no more reason than the small scale ones to deliver a convenient linear regime. How do you GP people check that your linear solutions and Born
approximation are valid? In petroleum applications, we use seismic surveys only when we suspect that severe heterogeneities (non Gaussian, non derivable, non tutti...) are present; otherwise who cares about surveying smooth surfaces?

3. Albert insists on accounting for the physics of the data (the waveform equations). I would agree if that physics is not obliterated by a wrong assumption about the physics of the media, that one called geology. There is no point to have one right if the other is abysmally wrong. Therefore, I would suggest repeating Albert’s forward modeling with a more realistic, hence non Gaussian, multi scale realization of the media and observe the impact of your various approximations, Born and tutti. This would be but one observation, completely dependent on the realization chosen, but it may tell something about robustness of these approximations.

Sorry Albert to bother you with old songs that you have heard umpteen times from Stanford.

Bests, Andre.

Email from AT 25-June-2007

Dear friends,

Yann Capdeville and I have been discussing about his ideas on the proper way to smooth a media, if necessary. And I am now convinced that Yann’s thoughts are not independent of some of the questions I have been raising for the waveform fitting problem.

Please, understand that some of the ideas here below are still in the confidential phase, until Yann duly publishes them.

To fix ideas, I shall consider an elastic, isotropic, one-dimensional medium. It may be made by a stack of a very large number of layers, or by a large one-dimensional grid, with one value of the elastic parameters \((\rho, \kappa, \mu)\) at each point of the grid. A sensible a priori distribution of models can, for instance, be introduced by using the logarithmic parameters \((\log \rho, \log \kappa, \log \mu)\) and assuming a Gaussian distribution. The mean of the distribution may be smooth (for instance a constant [logarithmic] gradient), and the covariance may be chosen such that the random realization have a nontrivial microstructure. If we work with a stack of layers, we deal with a covariance matrix, while if we work in the ”continuous” representation, we have a covariance function, but that is not essential for the discussion.

From this prior distribution, we choose one special random realization (by definition, if has microstructure), we invent a realistic source, we simulate some seismograms, we add to them some realistic noise, and call these the “observed” seismograms. It is assumed that we know the statistics of the noise (it may, for instance, also be a random realization of a Gaussian random field).

The inverse problem consists in using the “observed” seismograms to pass from the prior distribution of models to the posterior distribution. The theory for this is, I believe, that expressed in my view of images and reciprocal images of probabilities. There are two ways of using the theory, one is random but potentially exact, the other is deterministic but potentially biased. Let
us see the two of them in turn.

The first way is plain Monte Carlo: we should generate trillions of random realizations of the prior distribution, compute the predicted seismograms for each of the models, compare the predictions to the observations and, using the statistical description of the noise, give to each model a “chance of survival”. Then, it can be demonstrated that the surviving models (i.e., the models that have not been falsified by the observations) are samples of the posterior distribution. No potential bias here.

The second way comes from the knowledge that if the forward relation (model parameters -> seismograms) was exactly linear, the posterior distribution in the model space would be Gaussian (because the prior distribution was chosen Gaussian). If the forward relation is nonlinear, but not too far from being linear, then the posterior distribution is not Gaussian, but not too far from being Gaussian. The least-squares algorithm, then, is able to produce the center of the posterior distribution, and the covariance of the distribution that is tangent to the actual distribution (i.e., the covariance of the tangent Gaussian). The problem, of course, is that a serious analysis of how the nonlinearities map into non-gaussianities is far from trivial. From where a potential bias in the results.

In particular, looking at the detail of the least-squares optimization algorithm (for instance, a conjugate gradients algorithm or a square-root variable metric algorithm), we see that one initializes the algorithm at the center of the prior distribution (a smooth model) and one converges to a model that is the center of the posterior distribution (also a smooth model). So we have constantly been propagating waves in smooth models, although we definitely know that the true model has microstructure that, in some way, affected the “observed” seismograms. The least-squares algorithm, contrary to the plain Monte Carlo algorithm will never ask us to propagate waves in media with microstructure. From where my suspicion that there may be a price to pay for that in terms of potential biases or inaccuracies. And this fear seems to be confirmed when, after having obtained the mean of the posterior distribution, I evaluate the posterior covariance (so I have a perfect characterization of the posterior Gaussian) and I generate random realizations of this (approximate) posterior distributions. These model — that do contain microstructure — lead to residual seismograms that are too large to my taste.

Here enters Yann. He believes that the smooth model we obtain after the least-squares optimization is probably extremely close to the smoothed version of the true model, smoothed using the smoothing criteria he has developed (for instance, it is the quantity $1/c^2$ that has to averaged). This, of course, we can check in the numerical example I prepared for my teaching at Caltech, and that Qinya now masters.

Assume this is true. Then we arrive at the basic suggestion made by Yann (that I will write using my own inversion philosophy). Modify the forward modeling code using Capdeville’s smoothing theory (adapted to the spectrum of the source, and to the “spectrum” of the model parameters) (this implying nontrivial things, such as introducing new boundary conditions, or modifying the wave equation). The seismograms produced by this new theory will be arbitrarily accurate, yet cheap to calculate. Modify the a priori Gaussian distribution, say $G_{\text{prior}}$, by
smoothing it as it pleases to Yann. This gives some other distribution $G_{\text{prior}}'$. Solve the optimization problem using the modified (smoothed) theory and the modified (smoothed) distribution. This leads to a posterior distribution, say $G_{\text{post}}'$. Because we know the “filter” that transformed $G_{\text{prior}}$ into $G_{\text{prior}}'$, we can now (try to) use the inverse filter to pass from $G_{\text{post}}'$ into $G_{\text{post}}$, hopefully a good approximation of the actual posterior distribution, better than the one obtained via the elementary use of the least-squares theory.

This should be easy to check in the numerical example developed for my Caltech lessons.
References


