národní
úložiště
šedé
literatury

Recursive formulation of limited memory variable metric methods

Lukšan, Ladislav
2010

Dostupný z http://www.nusl.cz/ntk/nusl-41899

Dílo je chráněno podle autorského zákona č. 121/2000 Sb.

Tento dokument byl stažen z Národního úložiště šedé literatury (NUŠL).

Datum stažení: 07.06.2024

Další dokumenty můžete najít prostřednictvím vyhledávacího rozhraní nusl.cz .

Institute of Computer Science
Academy of Sciences of the Czech Republic

Recursive formulation of limited memory variable metric methods

Ladislav Lukšan, Jan Vlček

Technical report No. 1059

September 2010

Pod Vodárenskou věží 2, 182 07 Prague 8 phone: +420 2 688 42 44, fax: +420 2 858 57 89,
e-mail:e-mail:ics@cs.cas.cz

Institute of Computer Science
Academy of Sciences of the Czech Republic

Recursive formulation of limited memory variable metric methods

Ladislav Lukšan, Jan Vlček [1]

Technical report No. 1059

September 2010

Abstract:

In this report we propose a new recursive matrix formulation of limited memory variable metric methods. This approach enables to approximate of both the Hessian matrix and its inverse and can be used for an arbitrary update from the Broyden class (and some other updates). The new recursive formulation requires approximately $4mn$ multiplications and additions for the direction determination, so it is comparable with other efficient limited memory variable metric methods. Numerical experiments concerning Algorithm 1, proposed in this report, confirm its practical efficiency.

Keywords:
Unconstrained optimization, large scale optimization, limited memory methods, variable metric updates, recursive matrix formulation, algorithms.

[1] This work was supported by the Grant Agency of the Czech Republic, project No. 201/09/1957, and the institutional research plan No. AVOZ10300504

1 Introduction

Limited memory variable metric methods, introduced in [9], are intended for solving large scale unconstrained optimization problems with unknown or dense Hessian matrices. They are usually realized in a line search framework, so their iteration step has the form

$$x_{i+1} = x_i + t_i s_i \tag{1}$$

for $i \in \mathcal{N}$ ($\mathcal{N}$ is the set of positive integers), where $s_i = -H_i g_i$ is the direction vector (g_i is the gradient of the objective function and H_i is a positive definite approximation of the inverse Hessian matrix) and $t_i > 0$ is the step-length, which is taken to satisfy the weak Wolfe conditions

$$F_{i+1} - F_i \le \varepsilon_1 t_i s_i^T g_i, \tag{2}$$

$$s_i^T g_{i+1} \ge \varepsilon_2 s_i^T g_i, \tag{3}$$

with $0 < \varepsilon_1 < 1/2$ and $\varepsilon_1 < \varepsilon_2 < 1$. We restrict our attention to the limited memory variable metric methods from the Broyden class [7].

Let $0 < \bar{m} < n$, $i \in \mathcal{N}$ and $m = \min(\bar{m}, i)$. Limited memory variable metric methods from the Broyden class use direction vectors $s_1 = -g_1$ and $s_{i+1} = -H_{i+1} g_{i+1}$, $i \in \mathcal{N}$, where matrix $H_{i+1} \triangleq H_{i+1}^i$ is obtained from a sparse positive definite (usually scaled unit) matrix H_{i-m+1}^i by means of m updates

$$H_{j+1}^i = H_j^i + U_j^i M_j^i (U_j^i)^T, \tag{4}$$

$i - m + 1 \le j \le i$, where matrices $U_j^i = [d_j, H_j^i y_j]$ and M_j^i are chosen to satisfy quasi-Newton conditions $H_{j+1}^i y_j = d_j$, where $y_j = g_{j+1} - g_j$, $d_j = x_{j+1} - x_j$, $i - m + 1 \le j \le i$ (we use upper index i, to signify the relation to the i-th iteration). Formula (4) can be written in the form

$$H_{j+1}^i = H_j^i + \frac{1}{b_j} d_j d_j^T - \frac{1}{a_j^i} H_j^i y_j (H_j^i y_j)^T + \frac{\eta_j^i}{a_j^i} \left(\frac{a_j^i}{b_j} d_j - H_j^i y_j \right) \left(\frac{a_j^i}{b_j} d_j - H_j^i y_j \right)^T, \tag{5}$$

where $a_j^i = y_j^T H_j^i y_j$, $b_j = y_j^T d_j$ and η_j^i is a free parameter. Setting $\eta_j^i = 0$, $\eta_j^i = 1$ and $\eta_j^i = b_j/(b_j - a_j^i)$, we obtain the DFP, the BFGS and the Rank-1 updates, respectively. Note that the BFGS update is the most efficient one from these basic updates.

An advantage of limited memory variable metric methods described in this report is the fact that they can be realized in the way which requires (for n large) approximately $4mn$ multiplications and additions for the direction determination. Phrase approximately $4mn$ means that this number significantly dominates over additional required operations. For example, if $n = 1000$ and $m = 5$, then $4mn = 20000$, whereas $m^3 = 125$. There are two commonly used basic approaches: the recursive vector formulation based on the Strang recurrences [8] and the explicit matrix formulation proposed in [3]. To simplify the notation in the subsequent considerations, we will assume without the loss of generality that $i \le \bar{m}$. Then matrices (4) and (5) do not depend on the upper index, which can be omitted.

The first approach is applicable only in case all matrices H_j, $1 \le j \le i$, are obtained by the BFGS update (in fact there exists other possible updates realizable in this way, see [10], but they do not belong to the Broyden class). The recursive vector formulation of the limited memory BFGS method is based on the pseudo-product form: if $\eta_j = 1$, formula (5) can be written in the form

$$H_{j+1} = V_j^T H_j V_j + \frac{1}{b_j} d_j d_j^T, \qquad V_j = I - \frac{1}{b_j} y_j d_j^T. \tag{6}$$

Using this formula recursively, we obtain

$$H_{i+1} = \left(\prod_{j=1}^{i} V_j \right)^T H_1 \left(\prod_{j=1}^{i} V_j \right) + \sum_{k=1}^{i} \frac{1}{b_k} \left(\prod_{j=k+1}^{i} V_j \right)^T d_k d_k^T \left(\prod_{j=k+1}^{i} V_j \right).$$

Note that matrix H_{i+1} need not be stored, since vector $s_{i+1} = -H_{i+1} g_i$ can be obtained by two (Strang) recurrences. First we set $u_{i+1} = -g_{i+1}$ and compute numbers σ_j and vectors u_j, $i \geq j \geq 1$, by the backward recurrence

$$\sigma_j = \frac{d_j^T u_{j+1}}{b_j}, \qquad u_j = u_{j+1} - \sigma_j y_j. \tag{7}$$

Then we set $v_1 = H_1 u_1$ and compute vectors v_{j+1}, $1 \leq j \leq i$, by the forward recurrence

$$v_{j+1} = v_j + \left(\sigma_j - \frac{y_j^T v_j}{b_j} \right) d_j. \tag{8}$$

Finally we set $s_{i+1} = v_{i+1}$.

The use of the Strang recurrences (7)–(8) is the oldest (and simplest) possibility for implementing the limited memory BFGS method. As it was already mentioned, this approach is applicable only if all matrices H_j, $1 \leq j \leq i$, are obtained by the BFGS update. This disadvantage reveals when we need to update matrix $B_{i+1} = H_{i+1}^{-1}$. It follows from the duality (see [7]) that the Strang recurrences can be used only in case all matrices B_j, $1 \leq j \leq i$, are obtained by the DFP update. But the limited memory DFP method is much worse than the limited memory BFGS method, so this way is unsuitable.

The second approach is based on the fact that matrix H_{i+1}, obtained by recursive application of i updates of the form (4) to matrix H_1, can be written in the form

$$H_{i+1} = H_1 + \tilde{U}_i \tilde{M}_i \tilde{U}_i^T, \tag{9}$$

where $\tilde{U}_i = [d_1 - H_1 y_1, \ldots, d_i - H_1 y_i]$ and $\tilde{M}_i$ is a square matrix of order m for the Rank-1 update or $\tilde{U}_i = [d_1, \ldots, d_i, H_1 y_1, \ldots H_1 y_i]$ and $\tilde{M}_i$ is a square matrix of order $2m$ otherwise. For the basic updates (DFP, BFGS and Rank-1), the matrix $\tilde{M}_i$ can be expressed in the explicit form. Especially matrix H_{i+1}, obtained by recursive application of i BFGS updates to matrix H_1, can be written in the form

$$H_{i+1} = H_1 + [D_i, H_1 Y_i] \begin{bmatrix} (R_i^{-1})^T (C_i + Y_i^T H_1 Y_i) R_i^{-1}, & -(R_i^{-1})^T \\ -R_i^{-1}, & 0 \end{bmatrix} [D_i, H_1 Y_i]^T, \tag{10}$$

where $D_i = [d_1, \ldots, d_i]$, $Y_i = [y_1, \ldots, y_i]$, R_i is the i-dimensional upper triangular matrix such that $(R_i)_{kl} = d_k^T y_l$, $k \leq l$, $(R_i)_{kl} = 0$, $k > l$, and C_i is the i-dimensional diagonal matrix such that $(C_i)_{kk} = d_k^T y_k$ (see [3]). There exists a similar formula for matrix H_{i+1}, obtained by recursive application of i DFP updates to matrix H_1 (see [3]). Using the duality relation between the DFP and the BFGS updates, we can determine the matrix B_{i+1} obtained by recursive application of i BFGS updates to matrix B_1. The resulting matrix can be written in the form

$$B_{i+1} = B_1 - [Y_i, B_1 D_i] \begin{bmatrix} -C_i, & (L_i - C_i)^T \\ L_i - C_i, & D_i^T B_1 D_i \end{bmatrix}^{-1} [Y_i, B_1 D_i]^T, \tag{11}$$

where L_i is the i-dimensional lower triangular matrix such that $(L_i)_{kl} = d_k^T y_l$, $k \geq l$, $(L_i)_{kl} = 0$, $k < l$. The fact that we can use the inverse BFGS updates is very advantageous, since it allows us to implement variable metric trust region methods and methods for constrained optimization, which apply variable metric updates to the part of the KKT matrix.

In this report, we investigate a modification of the second approach. In Section 2, we propose a new recursive matrix formulation of limited memory variable metric methods. This approach can be used for both matrices H_{i+1} and B_{i+1} and for an arbitrary update from the Broyden class. Our recursive formulation requires approximately $4mn$ multiplications and additions for the direction determination, so it is comparable with the other approaches mentioned in this report. At the end of Section 2, we demonstrate that the recursive matrix formulation can be used for some other variable metric updates. As an example, we have chosen the Davidon class of variable metric updates proposed in [2] and reformulated in [5]. Section 3 contains results of numerical experiments which indicates that our approach is competitive with known limited memory variable metric methods.

2 The recursive matrix formulation

Let us assume that matrix H_{i+1} is obtained from matrix $H_1 = \lambda_i I$ by i updates of the form

$$H_{j+1} = H_j + U_j M_j U_j^T, \qquad 1 \leq j \leq i \tag{12}$$

(see (4)), where $U_j = [d_j, H_j y_j]$ and

$$M_j = \begin{bmatrix} \alpha_j, & \beta_j \\ \beta_j, & \gamma_j \end{bmatrix}.$$

We seek the expression

$$H_{i+1} = H_1 + \bar{U}_i \bar{M}_i \bar{U}_i^T, \tag{13}$$

where $\bar{U}_i = [d_1, H_1 y_1, \dots, d_i, H_1 y_i]$ and $\bar{M}_i$ is a square matrix of order $2m$. This formula is very similar to (9). For rank two updates, matrices $\bar{U}_i$ and $\tilde{U}_i$ differ only by orders of its columns. Note that the choice $H_1 = \lambda_i I$ (where usually $\lambda_i = d_i^T y_i / y_i^T y_i$) is essential for our considerations leading to the algorithm described below.

Theorem 1 *Let matrix H_{i+1} be obtained from matrix H_1 by i updates of the form (12). Then (13) holds with matrix $\bar{M}_i$ obtained recursively in such a way that $\bar{M}_1 = M_1$ and*

$$\bar{M}_j = \begin{bmatrix} \bar{M}_{j-1} + \gamma_j z_{j-1} z_{j-1}^T, & \beta_j z_{j-1}, & \gamma_j z_{j-1} \\ \beta_j z_{j-1}^T, & \alpha_j, & \beta_j \\ \gamma_j z_{j-1}^T, & \beta_j, & \gamma_j \end{bmatrix}, \qquad 2 \leq j \leq i, \tag{14}$$

where

$$z_{j-1} = \bar{M}_{j-1}\bar{r}_{j-1}, \qquad \bar{r}_{j-1} = \bar{U}_{j-1}^T y_j. \tag{15}$$

Proof We prove this theorem by induction. Assume that

$$H_j = H_1 + \bar{U}_{j-1}\bar{M}_{j-1}\bar{U}_{j-1}^T \tag{16}$$

for some index $2 \le j < i$. Relation (16) holds for $j = 2$ by (12) since $\bar{U}_1 = U_1$ and $\bar{M}_1 = M_1$. Substituting (16) into (12) and using the fact that

$$U_j = [d_j, H_j y_j] = \left[d_j,\, H_1 y_j + \bar{U}_{j-1}\bar{M}_{j-1}\bar{U}_{j-1}^T y_j\right] = \left[d_j,\, H_1 y_j + \bar{U}_{j-1} z_{j-1}\right]$$

by (15) and (16), we can write

$$\begin{aligned}
H_{j+1} &= H_1 + \bar{U}_{j-1}\bar{M}_{j-1}\bar{U}_{j-1}^T + \left[d_j,\, H_1 y_j + \bar{U}_{j-1} z_{j-1}\right] M_j \left[d_j,\, H_1 y_j + \bar{U}_{j-1} z_{j-1}\right]^T \\
&= H_1 + \bar{U}_{j-1}\, \bar{M}_{j-1}\, \bar{U}_{j-1}^T + \alpha_j\, d_j d_j^T \\
&\quad + \beta_j \left(d_j (H_1 y_j)^T + H_1 y_j d_j^T\right) + \beta_j \left(d_j\, (\bar{U}_{j-1} z_{j-1})^T + \bar{U}_{j-1} z_{j-1}\, d_j^T\right) \\
&\quad + \gamma_j\, H_1 y_j (H_1 y_j)^T + \gamma_j \left(H_1 y_j\, (\bar{U}_{j-1} z_{j-1})^T + \bar{U}_{j-1} z_{j-1}\, (H_1 y_j)^T\right) \\
&\quad + \gamma_j\, \bar{U}_{j-1} z_{j-1} z_{j-1}^T \bar{U}_{j-1}^T \\
&= H_1 + \left[\bar{U}_{j-1},\, d_j,\, H_1 y_j\right] \begin{bmatrix} \bar{M}_{j-1} + \gamma_j\, z_{j-1} z_{j-1}^T, & \beta_j\, z_{j-1}, & \gamma_j\, z_{j-1} \\ \beta_j\, z_{j-1}^T, & \alpha_j, & \beta_j \\ \gamma_j\, z_{j-1}^T, & \beta_j, & \gamma_j \end{bmatrix} \left[\bar{U}_{j-1},\, d_j,\, H_1 y_j\right]^T \\
&= H_1 + \bar{U}_j \bar{M}_j \bar{U}_j^T,
\end{aligned}$$

so the induction step is proved. □

Comparing (4) with (5), we can see that

$$\alpha_j = \frac{1}{b_j}\left(\eta_j \frac{a_j}{b_j} + 1\right), \qquad \beta_j = -\frac{\eta_j}{b_j}, \qquad \gamma_j = \frac{\eta_j - 1}{a_j}, \tag{17}$$

where $a_j = y_j^T H_j y_j$ and $b_j = y_j^T d_j$. Using (15) and (16), we obtain

$$a_j = y_j^T H_j y_j = y_j^T (H_1 y_j + \bar{U}_{j-1}\bar{M}_{j-1}\bar{U}_{j-1}^T y_j) = y_j^T H_1 y_j + \bar{r}_{j-1}^T z_{j-1},$$

so value a_j (required for the computation of α_j and γ_j by (17)) can be obtained by using known vectors $\bar{r}_{j-1}$ and z_{j-1}.

So far we have assumed that $1 \le i \le \bar{m}$. Now we describe the construction of matrix $H_{i+1} = \lambda_i I + \bar{U}_i \bar{M}_i \bar{U}_i^T$ in the general case. Let $m = \min(\bar{m}, i)$ and $S_i = \mathrm{diag}(1, \lambda_i, \ldots, 1, \lambda_i)$ (where $\lambda_i > 0$) be a $2m$-dimensional diagonal scaling matrix. Denote

$$\check{U}_{i-1} = [d_{i-m+1}, y_{i-m+1}, \ldots, d_{i-1}, y_{i-1}], \quad \check{R}_{i-1} = \begin{bmatrix} d_{i-m+1}^T y_{i-m+1}, & \ldots & d_{i-m+1}^T y_{i-1} \\ y_{i-m+1}^T y_{i-m+1}, & \ldots & y_{i-m+1}^T y_{i-1} \\ \ldots\ldots\ldots & \ldots & \ldots\ldots\ldots \\ 0, & \ldots & d_{i-1}^T y_{i-1} \\ 0, & \ldots & y_{i-1}^T y_{i-1} \end{bmatrix} \tag{18}$$

(these matrices are empty for $i = 1$) and

$$\hat{U}_i = [\check{U}_{i-1}, d_i, y_i], \qquad \hat{R}_i = \begin{bmatrix} \check{R}_{i-1}, & \check{U}_{i-1}^T y_i \\ 0, & d_i^T y_i \\ 0, & y_i^T y_i \end{bmatrix}. \tag{19}$$

Matrices $\check{R}_{i-1}$ and $\hat{R}_i$ are upper block triangular, where every block contains two rows and one column. Then $\bar{U}_i = S_i \hat{U}_i$ and matrix $\bar{M}_i \triangleq \hat{M}_i^i$ is obtained recursively in such a way that we set

$$\hat{M}_{i-m+1}^i = \begin{bmatrix} \alpha_{i-m+1}^i, & \beta_{i-m+1}^i \\ \beta_{i-m+1}^i, & \gamma_{i-m+1}^i \end{bmatrix} \tag{20}$$

and for $i - m + 1 \le j \le i - 1$ compute vector $z_j^i = \hat{M}_j^i S_j^i \hat{r}_j^i$, where S_j^i is the $2(j - i + m)$ dimensional leading submatrix of S_i and $\hat{r}_j^i$ is the $2(j - i + m)$ dimensional vector containing first $2(j - i + m)$ elements of the $(j - i + m)$-th column of matrix $\check{R}_{i-1}$, and set

$$\hat{M}_{j+1}^i = \begin{bmatrix} \hat{M}_j^i + \gamma_{j+1}^i \, z_j^i (z_j^i)^T, & \beta_{j+1}^i \, z_j^i, & \gamma_{j+1}^i \, z_j^i \\ \beta_{j+1}^i \, (z_j^i)^T, & \alpha_{j+1}^i, & \beta_{j+1}^i \\ \gamma_{j+1}^i \, (z_j^i)^T, & \beta_{j+1}^i, & \gamma_{j+1}^i \end{bmatrix}. \tag{21}$$

Using matrices obtained by the described way, direction vector s_{i+1} can be determined by the formula

$$s_{i+1} = -H_{i+1} g_{i+1} = -\lambda_i g_{i+1} - \bar{U}_i \bar{M}_i \bar{U}_i^T g_{i+1} = -\lambda_i g_{i+1} - \hat{U}_i S_i \hat{M}_i^i S_i \hat{U}_i^T g_{i+1}. \tag{22}$$

In this case, approximately $6mn$ multiplications and additions are consumed for the direction determination ($2mn$ for the determination of the last column of matrix $\hat{R}_i$ and $4mn$ for the computation of vector s_{i+1} by (22)) and approximately $2mn$ values are stored when n is large. Matrices $\check{U}_i$ and $\check{R}_i$ used in the next iteration are easily obtained from $\hat{U}_i$ and $\hat{R}_i$. If $i < \bar{m}$, then $\check{U}_i = \hat{U}_i$ and $\check{R}_i = \hat{R}_i$. If $i \ge \bar{m}$, then $\check{U}_i$ and $\check{R}_i$ arise from $\hat{U}_i$ and $\hat{R}_i$ after the deletion of the columns and rows depending on vectors with index $i - m + 1$. Thus

$$[d_{i-m+1}, y_{i-m+1}, \check{U}_i] = \hat{U}_i, \qquad \begin{bmatrix} d_{i-m+1}^T y_{i-m+1}, & [d_{i-m+1}^T y_{i-m+2}, \ldots, d_{i-m+1}^T y_i] \\ y_{i-m+1}^T y_{i-m+1}, & [y_{i-m+1}^T y_{i-m+2}, \ldots, y_{i-m+1}^T y_i] \\ 0, & \check{R}_i \end{bmatrix} = \hat{R}_i. \tag{23}$$

The above basic process can be modified in such a way that approximately $2mn$ multiplications and additions are dropped. As one can see from (21), the last column $\hat{r}_i$ of matrix $\hat{R}_i$ is not required for the computation of matrix $\hat{M}_i^i$. Thus we can compute vector $\hat{v}_i = \hat{U}_i^T g_{i+1}$ instead of $\hat{r}_i = \hat{U}_i^T y_i$. Vector $\hat{v}_i$ is then used for the determination of the direction vector by the formula

$$s_{i+1} = -\lambda_i g_{i+1} - \hat{U}_i S_i \hat{M}_i^i S_i \hat{v}_i. \tag{24}$$

After the determination of s_{i+1}, one can compute the first $2(m - 1)$ elements of $\hat{r}_i$ using the formula

$$\check{U}_{i-1}^T y_i = \check{U}_{i-1}^T g_{i+1} - \check{U}_{i-1}^T g_i, \tag{25}$$

where vector $\check{U}_{i-1}^T g_{i+1}$ contains the first $2(m-1)$ elements of $\hat{v}_i$ (see (19)) and vector $\check{U}_{i-1}^T g_i$ contains the last $2(m-1)$ elements of $\hat{v}_{i-1}$ (vector $\hat{v}_{i-1}$ is known from the previous iteration). The last two elements $d_i^T y_i$ and $y_i^T y_i$ of $\hat{r}_i$ are computed separately, since they serves for the determination of scaling parameter λ_i.

The above considerations are summarized in the following algorithm.

Algorithm 1 Data $\bar{m} < n$, $\underline{\varepsilon} > 0$, $0 < \varepsilon_1 < 1/2$, $\varepsilon_1 < \varepsilon_2 < 1$.

Step 1 Let $\check{U}_0$ and $\check{R}_0$ be empty matrices. Choose starting point $x_1 \in R^n$ and compute quantities $F_1 := F(x_1)$, $g_1 := g(x_1)$. Set $s_1 := -g_1$ and $i := 1$.

Step 2 If $\|g_i\| \le \underline{\varepsilon}$, terminate the computation, otherwise set $m := \min(\bar{m}, i)$.

Step 3 Determine step-size $t_i > 0$ satisfying conditions (2)–(3) and set $x_{i+1} := x_i + t_i s_i$. Compute new quantities $F_{i+1} := F(x_{i+1})$, $g_{i+1} := g(x_{i+1})$ and set $d_i := x_{i+1} - x_i$, $y_i := g_{i+1} - g_i$. Compute values $d_i^T y_i$, $y_i^T y_i$ and set $\lambda_i := d_i^T y_i / y_i^T y_i$ to define $2m$ dimensional scaling matrix $S_i := \mathrm{diag}(1, \lambda_i, \dots, 1, \lambda_i)$.

Step 4 Determine matrix $\hat{M}_{i-m+1}^i$ by formula (20). Set $\hat{U}_i := [\check{U}_{i-1}, d_i, y_i]$, $\hat{v}_i := \hat{U}_i^T g_{i+1}$ and $j := i - m + 1$.

Step 5 If $j = i$ go to Step 7.

Step 6 Choose the value of parameter η_j^i appearing in (17). Set $z_j^i := \hat{M}_j^i S_j^i \check{r}_j^i$, where S_j^i is the $2(j-i+m)$ dimensional leading submatrix of S_i and $\check{r}_j^i$ is the $2(j-i+m)$ dimensional vector containing the first $2(j-i+m)$ elements of the $(j-i+m)$-th column of matrix $\check{R}_{i-1}$, compute matrix $\hat{M}_{j+1}^i$ by (21), set $j := j + 1$ and go to Step 5.

Step 7 Set $\bar{M}_i := \hat{M}_i^i$ and compute direction vector s_{i+1} by formula (24). Compute vector $\check{U}_{i-1}^T y_i$ by (25) and matrix $\hat{R}_i$ by (19).

Step 8 If $i < \bar{m}$, set $\check{U}_i := \hat{U}_i$ and $\check{R}_i := \hat{R}_i$, otherwise determine $\check{U}_i$ and $\check{R}_i$ by (23). Set $i := i + 1$ and go to Step 2.

The recursive matrix formulation described above can be used also for some other variable metric updates. We focus our attention on the Davidon class of variable metric methods proposed in [2] and reformulated in [5]. Variable metric methods from this class are generalizations of the Rank-1 method. Applied to the quadratic function, they generate conjugate directions without perfect line search.

Limited memory variable metric methods from the Davidon class generate matrix H_{i+1} from matrix $H_1 = \lambda_i I$ by i updates of the form

$$H_{j+1} = H_j + V_j N_j V_j^T, \qquad 1 \le j \le i, \tag{26}$$

where $V_j = [v_j, d_j - H_j y_j]$ and

$$N_j = \begin{bmatrix} \rho_j, & \sigma_j \\ \sigma_j, & \tau_j \end{bmatrix}.$$

Vector v_j is generated recursively to satisfy conditions

$$v_{j+1} \in \mathrm{span}(v_j, d_j - H_j y_j), \qquad v_{j+1}^T y_j = 0 \tag{27}$$

(vector v_{j+1} is a linear combination of vectors v_j, $d_j - H_j y_j$ and is perpendicular to vector y_j). Conditions (27) are satisfied, e.g., if

$$v_{j+1} = y_j^T(d_j - H_j y_j)v_j - y_j^T v_j(d_j - H_j y_j). \tag{28}$$

It can be easily proved, see [5], that the update $H_{j+1} = H_j + V_j N_j V_j^T$, where $V_j = [v_j, d_j - H_j y_j]$, satisfies quasi-Newton condition $H_{j+1} y_j = d_j$, if

$$H_{j+1} = H_j + \frac{(d_j - H_j y_j)(d_j - H_j y_j)^T}{y_j^T(d_j - H_j y_j)} - \frac{\varphi_j v_{j+1} v_{j+1}^T}{y_j^T(d_j - H_j y_j)}, \tag{29}$$

where $\varphi_j = -\det N_j$ is a free parameter and v_{j+1} is the vector determined by formula (28). Thus

$$\rho_j = -\varphi_j y_j^T(d_j - H_j y_j), \qquad \sigma_j = \varphi_j y_j^T v_j, \qquad \tau_j = \frac{1 - \varphi_j (y_j^T v_j)^2}{y_j^T(d_j - H_j y_j)}. \tag{30}$$

Setting $\varphi_j = 0$, we obtain the Rank-1 update which lies in both the Broyden and the Davidon classes. It is important that some updates from the Davidon class generate positive definite matrices, but it is computationally difficult to find a suitable value of parameter φ_j, see [5]. Notice that we have chosen the Davidon class of variable metric updates not for its efficiency, but for the demonstration of the fact that the recursive matrix formulation can be also used for variable metric updates that do not belong to the Broyden class.

Analogously to (13), we seek the expression

$$H_{i+1} = H_1 + \bar{V}_i \bar{N}_i \bar{V}_i^T, \tag{31}$$

where $\bar{V}_i = [v_1, d_1 - H_1 y_1, \dots, v_i, d_i - H_1 y_i]$ and $\bar{N}_i$ is a square matrix of order $2m$.

Theorem 2 *Let matrix H_{i+1} be obtained from matrix H_1 by i updates of the form (26). Then (31) holds with matrix $\bar{N}_i$ obtained recursively in such a way that $\bar{N}_1 = N_1$ and*

$$\bar{N}_j = \begin{bmatrix} \bar{N}_{j-1} + \tau_j\, z_{j-1} z_{j-1}^T, & \sigma_j\, z_{j-1}, & \tau_j\, z_{j-1} \\ \sigma_j\, z_{j-1}^T, & \rho_j, & \sigma_j \\ \tau_j\, z_{j-1}^T, & \sigma_j, & \tau_j \end{bmatrix}, \qquad 2 \le j \le i, \tag{32}$$

where

$$z_{j-1} = \bar{N}_{j-1} \bar{r}_{j-1}, \qquad \bar{r}_{j-1} = \bar{V}_{j-1}^T y_j. \tag{33}$$

Proof We prove this theorem by induction. Assume that

$$H_j = H_1 + \bar{V}_{j-1} \bar{N}_{j-1} \bar{V}_{j-1}^T \tag{34}$$

for some index $2 \le j < i$. Relation (34) holds for $j = 2$ by (26) since $\bar{V}_1 = V_1$ and $\bar{N}_1 = N_1$. Denoting $w_j = d_j - H_1 y_j$, substituting (34) into (26) and using the fact that

$$V_j = [v_j, d_j - H_j y_j] = \left[v_j,\, d_j - H_1 y_j + \bar{V}_{j-1} \bar{N}_{j-1} \bar{V}_{j-1}^T y_j\right] = \left[v_j,\, w_j + \bar{V}_{j-1} z_{j-1}\right]$$

by (33) and (34), we can write

$$\begin{aligned}
H_{j+1} &= H_1 + \bar{V}_{j-1}\bar{N}_{j-1}\bar{V}_{j-1}^T + \left[v_j,\, w_j + \bar{V}_{j-1}z_{j-1}\right] N_j \left[v_j,\, w_j + \bar{V}_{j-1}z_{j-1}\right]^T \\
&= H_1 + \bar{V}_{j-1}\,\bar{N}_{j-1}\,\bar{V}_{j-1}^T + \rho_j\, v_j v_j^T \\
&\quad + \sigma_j \left(v_j w_j^T + w_j v_j^T\right) + \sigma_j \left(v_j\,(\bar{V}_{j-1}z_{j-1})^T + \bar{V}_{j-1}z_{j-1}\, v_j^T\right) \\
&\quad + \tau_j\, w_j w_j^T + \tau_j \left(w_j\,(\bar{V}_{j-1}z_{j-1})^T + \bar{V}_{j-1}z_{j-1}\, w_j^T\right) \\
&\quad + \tau_j\, \bar{V}_{j-1}z_{j-1}z_{j-1}^T\bar{V}_{j-1}^T \\
&= H_1 + \left[\bar{V}_{j-1},\, v_j,\, w_j\right] \left[\begin{array}{lll} \bar{N}_{j-1} + \tau_j\, z_{j-1}z_{j-1}^T, & \sigma_j\, z_{j-1}, & \tau_j\, z_{j-1} \\ \sigma_j\, z_{j-1}^T, & \rho_j, & \sigma_j \\ \tau_j\, z_{j-1}^T, & \sigma_j, & \tau_j \end{array}\right] \left[\bar{V}_{j-1},\, v_j,\, w_j\right]^T \\
&= H_1 + \bar{V}_j\bar{N}_j\bar{V}_j^T,
\end{aligned}$$

so the induction step is proved. □

Using (33) and (34), we obtain

$$d_j - H_j y_j = d_j - H_1 y_j + \bar{V}_{j-1}\bar{M}_{j-1}\bar{V}_{j-1}^T y_j = d_j - H_1 y_j + \bar{V}_{j-1}z_{j-1},$$

and

$$y_j^T(d_j - H_j y_j) = y_j^T d_j - y_j^T H_1 y_j + \bar{r}_{j-1}^T z_{j-1}.$$

These quantities are necessary for the determination of vector v_j by (28) and for the computation of numbers ρ_j, σ_j, τ_j by (32).

3 Numerical experiments and conclusions

Limited memory variable metric methods from the Broyden class were tested by using 72 unconstrained minimization problems with 1000 variables from the collection TEST25 described in [6] (ten problems 48, 57–58, 60–61, 67–70, 79, which are unsuitable for testing limited memory variable metric methods, were excluded). This collections can be found on `http://www.cs.cas.cz/luksan/test.html` together with report [6]. The results of these tests are presented in Table 1, where `NIT` is the total number of iterations, `NFV` is the total number of function evaluations, `Fail` is the total number of failures and `Time` is the total computational time. Note that the total computational time is not always proportional to the total number of function evaluations, since individual test problems have different complexity. Table 1 contains two sets of columns corresponding to limited memory methods with $\bar{m} = 5$ and $\bar{m} = 10$, respectively. Rows are partitioned into 3 groups. The first group corresponds to the new limited memory variable metric method (Algorithm 1) with various constant values of parameter η. The second group contains results obtained by Algorithm 1 with two special choices of parameter η: `H` – the Hoshino update proposed in [4], for which $\eta = b/(b+a)$, and `N` – the update proposed in [7], for which

$$\begin{aligned}
\eta &= \frac{\max(0, \sqrt{c/a} - b^2/(ac))}{1 - b^2/(ac)}, && b^2/(ac) < 1, \\
\eta &= 1, && b^2/(ac) \ge 1.
\end{aligned}$$

The third group introduces comparison of three versions of the limited memory BFGS method: RV – recursive vector formulation (using Strang recurrences), EM – explicit matrix formulation (using matrix (10)) and RM – recursive matrix formulation (Algorithn 1). For implementing all the above mentioned methods, we have used the same line search subroutine with parameters $\underline{\varepsilon} = 10^{-6}$, $\varepsilon_1 = 0.001$, $\varepsilon_1 = 0.9$.

	$\bar{m} = 5$				$\bar{m} = 10$			
Method	NIT	NFV	Fail	Time	NIT	NFV	Fail	Time
$\eta = 0.6$	129825	131874	–	36.55	139660	141900	–	50.30
$\eta = 0.8$	123958	127862	–	34.92	133975	138004	–	47.78
$\eta = 1.0$	126167	132279	–	36.22	123850	129890	–	42.57
$\eta = 1.2$	118404	126631	–	33.70	131783	139987	–	46.75
$\eta = 1.4$	118818	130306	–	34.70	129372	141227	–	48.50
$\eta = 1.6$	121316	136657	–	37.99	131229	149917	–	47.05
H	185025	186126	1	50.30	153603	154596	–	53.95
N	129711	137764	–	38.03	124617	133829	–	44.25
BFGS-RV	123699	129568	–	36.92	130067	135933	–	45.16
BFGS-EM	122491	128527	–	36.33	129723	135726	–	46.14
BFGS-RM	126167	132279	–	36.22	123850	129890	–	42.57

Table 1

From the results presented in Table 1, we can deduce that limited memory variable metric methods with the recursive matrix formulation are competitive with other realizations of limited memory variable metric methods (they use approximately $4mn$ operations for the direction determination as well). The BFGS update seems to be the best one from the Broyden class within the limited memory framework (even if, for $\bar{m} = 5$, the choice $\eta = 1.2$ gave better results). Since we have tested only a limited number of simple updates, it is possible that a more successful choice of parameter η will be found. It is important to say that such an update can be realized by our recursive formulation approach.

References

[1] I. Bongartz, A.R. Conn, N. Gould, P.L. Toint: CUTE: constrained and unconstrained testing environment, ACM Transactions on Mathematical Software 21 (1995), 123-160.

[2] W.C.Davidon: Optimally conditioned optimization algorithms without line searches. Mathematical Programming 9 (1975) 1-30.

[3] R.H.Byrd, J.Nocedal, R.B.Schnabel: Representation of quasi-Newton matrices and their use in limited memory methods. Mathematical Programming 63 (1994) 129-156.

[4] S.Hoshino: A formulation of variable metric methods. Journal of Institute of Mathematics and its Applications 10 (1972) 394-403.

[5] L.Lukšan: Quasi-Newton methods without projections for unconstrained minimization. Kybernetika 18 (1982) 290-306.

[6] L.Lukšan, C.Matonoha, J.Vlček: Sparse test problems for unconstrained optimization. Report V-1064, Institute of Computer Science AS CR, Prague, 2010.

[7] L.Lukšan, E.Spedicato: Variable metric methods for unconstrained optimization and nonlinear least squares. Journal of Computational and Applied Mathematics 124 (2000) 61-95.

[8] H.Matthies, G.Strang: The solution of nonlinear finite element equations. International Journal for Numerical Methods in Engineering 14 (1979) 1613-1623.

[9] J.Nocedal: Updating quasi-Newton matrices with limited storage. Mathematics of Computation 35 (1980) 773-782.

[10] J.Vlček, L.Lukšan: Generalizations of the limited-memory BFGS method based on quasi-product form of update. Report V-1060, Institute of Computer Science AS CR, Prague 2009.