LSA: Description of methods

1. Introduction

This document contains a detailed description of methods implemented in subroutines PNET, PNED, PNEC, PSED, PSEC, PSEN, PGAD, PGAC, PMAX, PSUM, PEQN, PEQL.

2. Matrix-free methods for general problems

Consider a general twice continuously differentiable function $F : R^n \to R$, where n is large, and assume that the structure of the Hessian matrix of F is unknown. In this case, subroutines PLIS, PLIP, and PNET based on matrix-free methods can be used for seeking a local minimum of F. These methods are realized in the line-search framework so that they generate a sequence of points $x_k \in R^n$, $k \in N$, by the simple process

$$x_{k+1} = x_k + \alpha_k d_k, \tag{1}$$

where $d_k \in R^n$ is a direction vector and $0 < \alpha_k \le \overline{\alpha}_k$ is a scalar step-size. The direction vector is determined in such a way that

$$-d_k^T g(x_k) \ge \underline{\varepsilon} \|d_k\| \|g(x_k)\| \tag{2}$$

(the uniform descent condition) holds, where $g(x_k)$ is the gradient of F at the point x_k and $0 < \underline{\varepsilon} \le 1$ (we use the value $\underline{\varepsilon} = 10^{-4}$ in our subroutines). The step-size is chosen in such a way that

$$F(x_{k+1}) - F(x_k) \le \varepsilon_1 \, \alpha_k \, d_k^T g(x_k), \quad d_k^T g(x_{k+1}) \ge \varepsilon_2 \, d_k^T g(x_k) \tag{3}$$

(the weak Wolfe conditions) hold, where $0 < \varepsilon_1 < 1/2$ is a tolerance for the function value decrease and $\varepsilon_1 < \varepsilon_2 < 1$ is a tolerance for the directional derivative increase (we use the values $\varepsilon_1 = 0.0001$ and $\varepsilon_2 = 0.9$ in our subroutines). The maximum step-size $\overline{\alpha}_k$ is given by the formula $\overline{\alpha}_k = \overline{\Delta}/\|d_k\|$, where $\overline{\Delta}$ is an upper bound for the norm $\|x_{k+1} - x_k\|$ (parameter XMAX in our subroutines). The step-size α_k is chosen iteratively either by bisection (MES $= 1$), or by quadratic interpolation with two function values (MES $= 2$), or by quadratic interpolation with two directional derivatives (MES $= 3$), or by cubic interpolation (MES $= 4$). We start with the initial estimate $\alpha_k = 1$ if IEST $= 0$ or the initial estimate is derived by using the lower bound for F (parameter FMIN in our subroutine) if IEST $= 1$.

The direction vector d_k is usually computed by the formula $d_k = -H_k g_k$ or by solving the linear system $B_k d_k = -g_k$, where $g_k = g(x_k)$, B_k is an approximation of the Hessian matrix $G(x_k)$, and H_k is an approximation of its inverse. Limited-memory variable metric methods use the matrix H_k implicitly (it is not stored). Similarly, the truncated Newton method uses the matrix B_k, which is not stored as well. If the direction vector computed by using the above way does not satisfy the uniform descent condition (2), a restart is performed, which implies that the direction vector $d_k = -g_k$, satisfying (2), is used.

2.1 Limited-memory BFGS method

Subroutine PLIS is an implementation of the limited-memory BFGS method proposed in [13], [26]. This method works with matrices $H_k = H_k^k$, where $H_{k-m}^k = \gamma_k I$, $\gamma_k > 0$ and

$$H_{j+1}^k = V_j^T H_j^k V_j + \frac{1}{b_j} s_j s_j^T, \qquad V_j = I - \frac{1}{b_j} y_j s_j^T$$

for $k - m \le j \le k - 1$. Here $s_j = x_{j+1} - x_j$, $y_j = g_{j+1} - g_j$, $a_j = y_j^T H_j^k y_j$, $b_j = y_j^T s_j$. Thus

$$H_{j+1}^k = \frac{b_{k-1}}{a_{k-1}} \left(\prod_{i=k-m}^{j} V_i \right)^T \left(\prod_{i=k-m}^{j} V_i \right) + \sum_{l=k-m}^{j} \frac{1}{b_l} \left(\prod_{i=l+1}^{j} V_i \right)^T s_l s_l^T \left(\prod_{i=l+1}^{j} V_i \right)$$

(we use $\gamma_k = b_{k-1}/a_{k-1}$ in our implementation). The matrix $H_k = H_k^k$ need not be constructed explicitly since we need only a vector $d_k = -H_k^k g_k$, which can be computed by using two recurrences (the Strang formula). First, vectors

$$u_j = -\left(\prod_{i=j}^{k-1} V_i\right) g_k,$$

$k-1 \geq j \geq k-m$, are computed by using the backward recurrence

$$\begin{aligned} \sigma_j &= s_j^T u_{j+1}/b_j, \\ u_j &= u_{j+1} - \sigma_j y_j, \end{aligned}$$

where $u_k = -g_k$. Then vectors

$$v_{j+1} = \frac{b_{k-1}}{a_{k-1}} \left(\prod_{i=k-m}^{j} V_i\right)^T u_{k-m} + \sum_{l=k-m}^{j} \frac{1}{b_l} \left(\prod_{i=l+1}^{j} V_i\right)^T s_l s_l^T u_{l+1},$$

$k-m \leq j \leq k-1$, are computed by using the forward recurrence

$$v_{j+1} = v_j + (\sigma_j - y_j^T v_j/b_j) s_j,$$

where $v_{k-m} = (b_{k-1}/a_{k-1}) u_{k-m}$. Finally, we set $d_k = v_k$. Note that $2m$ vectors s_j, y_j, $k-m \leq j \leq k-1$ are used and stored. The number of consecutive variable metric updates is defined as $m = \min(\mathtt{MF}, k - \underline{k})$, where MF is a parameter of the subroutine PLIS and $\underline{k}$ is an index of the iteration corresponding to the last restart.

2.2 Shifted limited-memory variable metric methods

Subroutine PLIP is an implementation of shifted limited-memory variable metric methods proposed in [?]. These methods work with matrices $H_k = \zeta_k I + U_k U_k^T$, where $n \times m$ matrix U_k is updated by formula $U_{k+1} = V_k U_k$ with a low rank matrix V_k chosen in such a way that the (modified) quasi-Newton condition $U_{k+1} U_{k+1}^T y_k = \rho_k \tilde{s}_k$ with $\tilde{s}_k = s_k - \zeta_{k+1} y_k$ is satisfied (we use the same notation, namely s_k, y_k, a_k, b_k as in Section 2.1). This condition can be replaced by equations

$$U_{k+1}^T y_k = z_k, \qquad U_{k+1} z_k = \rho_k \tilde{s}_k, \qquad \|z_k\|^2 = \rho_k y_k^T \tilde{s}_k.$$

where z_k is an optional vector parameter. Note that the last equality, which is a consequence of the first two equalities, is the only restriction laid on the vector z_k. To simplify the notation, we define vectors $u_k = U_k^T y_k$ and $v_k = U_k^T H_k^{-1} s_k = -\alpha_k U_k^T g_k$.

The choice of a shift parameter ζ_{k+1} is a crucial part of shifted limited-memory variable metric methods. The value

$$\zeta_{k+1} = \mu_k \frac{b_k}{\|y_k\|^2}, \qquad \mu_k = \frac{\sqrt{1 - \|u_k\|^2/a_k}}{1 + \sqrt{1 - b_k^2/(\|s_k\|^2 \|y_k\|^2)}}$$

is used in subroutine PLIP. The most efficient shifted limited-memory variable metric methods can be derived by a variational principle. Let T_k be a symmetric positive definite matrix. It can be shown (see [?]) that the Frobenius norm $\|T_k^{-1/2}(U_{k+1} - U_k)\|_F^2$ is minimal on the set of all matrices satisfying the quasi-Newton condition if and only if

$$U_{k+1} = U_k - \frac{T_k y_k}{y_k^T T_k y_k} y_k^T U_k + \left(\rho_k \tilde{s}_k - U_k z_k + \frac{y_k^T U_k z_k}{y_k^T T_k y_k} T_k y_k\right) \frac{z_k^T}{\|z_k\|^2}.$$

Here $T_k y_k$ and z_k are vector parameters defining a class of shifted limited-memory variable metric methods. Using suitable values of these vectors, we obtain particular methods of this class.

Assuming that $T_k y_k$ and $\rho_k \tilde{s}_k - U_k z_k$ are linearly dependent and setting

$$z_k = \vartheta_k v_k, \qquad \vartheta_k = \pm\sqrt{\rho_k y_k^T \tilde{s}_k / \|v_k\|^2}$$

we obtain rank 1 variationally derived method (VAR1), where

$$U_{k+1} = U_k - \frac{\rho_k \tilde{s}_k - \vartheta_k U_k v_k}{\rho_k y_k^T \tilde{s}_k - \vartheta_k u_k^T v_k} (u_k - \vartheta_k v_k)^T ,$$

which gives the best results for the choice $\mathrm{sgn}(\vartheta_k u_k^T v_k) = -1$. Method VAR1 is chosen if $\mathtt{MET} = 1$ in the subroutine `PLIP`. Using z_k given above and setting $T_k y_k = \tilde{s}_k$, which corresponds to the BFGS method in the full-memory case, we obtain rank 2 variationally derived method (VAR2), where

$$U_{k+1} = U_k - \frac{\tilde{s}_k}{y_k^T \tilde{s}_k} u_k^T + \left(\rho_k \frac{\tilde{s}_k}{\vartheta_k} - U_k v_k + \frac{u_k^T v_k}{y_k^T \tilde{s}_k} \tilde{s}_k \right) \frac{v_k^T}{\|v_k\|^2}.$$

Method VAR2 is chosen if $\mathtt{MET} = 2$ in the subroutine `PLIP`. The efficiency of both these methods depends on the value of the correction parameter ρ_k. This value is determined by the formula $\rho_k = \sqrt{\nu_k \varepsilon_k}$. Here $\nu_k = \mu_k/(1-\mu_k)$, μ_k is a relative shift parameter defined above and

$$\varepsilon_k = \sqrt{1 - \|u_k\|^2/a_k}$$

is a damping factor of μ_k. The number of columns of the matrix U_k is defined as $m = \min(\mathtt{MF}, k - \underline{k})$, where `MF` is a parameter of the subroutine `PLIP` and $\underline{k}$ is an index of the iteration corresponding to the last restart.

2.3 Inexact truncated Newton method

Subroutine `PNET` is based on a line-search realization of the Newton method, which uses conjugate gradient (CG) iterations for solving a system of linear equations $G(x)d = -g(x)$ ($g(x)$ and $G(x)$ are the gradient and the Hessian matrix of the function $F : R^n \to R$ at the point x, respectively). Since the matrix $G(x)$ can be indefinite, a modification of the (possibly preconditioned) CG method is used, which terminates if a negative curvature is detected. More precisely, we set $d_1 = 0$, $g_1 = g(x)$, $h_1 = C^{-1} g_1$, $p_1 = -h_1$, $\sigma_1 = g_1^T h_1$ and for $i = 1, 2, 3, \dots$ we proceed in the following way. If $\|g_i\| \le \overline{\omega}(x)\|g(x)\|$, where $\overline{\omega}(x)$ is a local precision (see (5)), then we set $d = d_i$ and terminate the computation, else we set

$$q_i = G(x) p_i, \qquad \tau_i = p_i^T q_i.$$

If $\tau_i < \underline{\tau}$ ($\underline{\tau}$ is the lowest permitted local curvature), then we set $d = -g(x)$ (if $i = 1$) or $d = d_i$ (if $i > 1$) and terminate the computation, else we set $\alpha_i = \sigma_i/\tau_i$ and compute

$$\begin{aligned} d_{i+1} &= d_i + \alpha_i p_i, \qquad g_{i+1} = g_i + \alpha_i q_i, \\ h_{i+1} &= C^{-1} g_{i+1}, \qquad \sigma_{i+1} = g_{i+1}^T h_{i+1}, \\ p_{i+1} &= -h_{i+1} + (\sigma_{i+1}/\sigma_i) p_i. \end{aligned}$$

The principal feature of the inexact truncated Newton method is the fact that the Hessian matrix $G(x)$ is not used explicitly, but the vector $G(x)p_i$ is computed by numerical differentiation using the formula

$$G(x) p_i \approx \frac{g(x + \delta_i p_i) - g(x)}{\delta_i}, \tag{4}$$

where $\delta_i = \sqrt{\varepsilon_M}/\|p_i\|$ (ε_M is the machine precision). Thus one extra gradient evaluation is needed in every CG iteration.

The CG method is terminated if $\tau_i < \underline{\tau}$ (a small curvature is detected) or if $\|g_i\| \le \overline{\omega}(x)\|g(x)\|$ (a sufficient precision is achieved). We use the value $\underline{\tau} = 10^{-60}$ in subroutine `PNET`. The value $\overline{\omega}(x)$ is chosen according to the inexact Newton approach [7]. In the k-th Newton iteration we use the value

$$\overline{\omega}(x_k) = \min\left(\sqrt{\|g(x_k)\|}, 1/k, \overline{\omega} \right), \tag{5}$$

where $\overline{\omega} = 0.8$ in subroutine `PNET`.

The matrix C serving as a preconditioner (symmetric and positive definite) cannot be derived from the Hessian matrix, which is not known explicitly. If `MOS2` $= 1$ (`MOS2` is a parameter of the subroutine `PNET`), the unpreconditioned CG method ($C = I$) is used. If `MOS2` $= 2$, we use the preconditioner obtained by the limited memory BFGS method. In this case $C^{-1} = H_k = H_k^k$, where H_k^k is a matrix defined in Subsection 2.1. This matrix need not be constructed explicitly, since we need only the vector $h_i = C^{-1}g_i = H_k^k g_i$ (i is an inner index of the CG method), which can be computed by using two recurrences (the Strang formula). First, vectors u_j, $k-1 \geq j \geq k-m$, are computed by using the backward recurrence

$$\begin{aligned} \sigma_j &= s_j^T u_{j+1}/b_j, \\ u_j &= u_{j+1} - \sigma_j y_j, \end{aligned}$$

where $u_k = g_i$. Then vectors v_{j+1}, $k-m \leq j \leq k-1$, are computed by using the forward recurrence

$$v_{j+1} = v_j + (\sigma_j - y_j^T v_j/b_j)s_j,$$

where $v_{k-m} = (b_{k-1}/a_{k-1})u_{k-m}$. Finally, we set $h_i = v_k$. Note that $2m$ additional vectors $s_j = x_{j+1} - x_j$, $y_j = g_{j+1} - g_j$, $k-m \leq j \leq k-1$, has to be stored if `MOS2` $= 2$.

2.4 Active set strategy for box constraints

If box constraints are considered, then a simple active set strategy is used. To simplify the notation, we omit the iteration index k in the following description. Every iteration is started by the detection of new candidates for active constraints. Thus we set

$$\begin{array}{rlcll} x_i = x_i^l, & I_i^x = -1 & \text{if} & I_i^x = 1, & x_i \leq x_i^l + \varepsilon_c \max(|x_i^l|, 1), \\ x_i = x_i^u, & I_i^x = -2 & \text{if} & I_i^x = 2, & x_i \geq x_i^u - \varepsilon_c \max(|x_i^u|, 1), \\ x_i = x_i^l, & I_i^x = -3 & \text{if} & I_i^x = 3, & x_i \leq x_i^l + \varepsilon_c \max(|x_i^l|, 1), \\ x_i = x_i^u, & I_i^x = -4 & \text{if} & I_i^x = 3, & x_i \geq x_i^u - \varepsilon_c \max(|x_i^u|, 1), \\ & I_i^x = -5 & \text{if} & I_i^x = 5 & \end{array}$$

for $1 \leq i \leq n$, where ε_c is a required precision (we use the value $\varepsilon_c = 10^{-8}$ in our subroutines). After computing gradient $g = g(x)$, we determine a projected gradient g^p and a chopped gradient g^c in such a way that

$$\begin{array}{rlcl} g_i^p = 0, & g_i^c = \min(0, g_i) & \text{for} & I_i^x = -1 \quad \text{or} \quad I_i^x = -3, \\ g_i^p = 0, & g_i^c = \max(0, g_i) & \text{for} & I_i^x = -2 \quad \text{or} \quad I_i^x = -4, \\ g_i^p = 0, & g_i^c = 0 & \text{for} & I_i^x = -5, \\ g_i^p = g_i, & g_i^c = 0 & \text{for} & I_i^x \geq 0. \end{array}$$

If $\|g^c\|_\infty > \|g^p\|_\infty$ and the previous step was successful, we delete insubstantial active constraints by setting

$$\begin{array}{lcccc} I_i^x = 1 & \text{if} & I_i^x = -1 & \text{and} & g_i < 0, \\ I_i^x = 2 & \text{if} & I_i^x = -2 & \text{and} & g_i > 0, \\ I_i^x = 3 & \text{if} & I_i^x = -3 & \text{and} & g_i < 0, \\ I_i^x = 3 & \text{if} & I_i^x = -4 & \text{and} & g_i > 0. \end{array}$$

In this way, we have obtained a current set of active constraints for the direction determination, step-size selection and variable metric update. This active set defines a reduced problem with variables x_i, $I_i^x < 0$, fixed. If we introduce a matrix Z containing columns e_i, $I_i^x \geq 0$ (e_i is the i-th column of the unit matrix), we can define a reduced gradient $g^r = Z^T g$ and reduced matrices H^r or B^r to obtain reduced direction vectors $d^r = -H^r g^r$ or $B^r d^r = -g^r$. Finally, we use the direction vector $d = Zd^r$. In this way, we can adapt an arbitrary line-search or trust-region method to solve the reduced problem. Since the set of active constraints can change, we have to use a suitable restart strategy (conditions for setting $B^r = I$, $H_r = I$ or $d^r = -g^r$). To guarantee a descent, the restart is always performed when more then one insubstantial active constraint is deleted. The preconditioned truncated Newton method is restarted after every change of the set of active constraints.

¿From a practical point of view, it is not advantageous to construct reduced quantities. A better way is to construct projected gradients $g^p = ZZ^T g$ and projected matrices $H^p = ZZ^T HZZ^T + YY^T$ or $B^p = ZZ^T BZZ^T + YY^T$, where Y contains columns e_i, $I_i^x < 0$, to obtain direction vectors $d = -H^p g^p$ or $B^p d = -g^p$. Matrices H^p or B^p are block diagonal (with blocks H^r, I or B^r, I) and they can be updated by using projected vectors $s^p = ZZ^T s = ZZ^T(x^+ - x)$ and $y^p = ZZ^T(g^+ - g)$. Thus it suffices to use projected quantities instead of standard ones. Diagonal blocks of matrices H^p or B^p can be easily derived from their sparsity pattern by considering only the elements H_{ij}^p or B_{ij}^p for which $I_i^x \geq 0$, $I_j^x \geq 0$ hold simultaneously. These blocks are then used in matrix multiplications and matrix decompositions.

Before the step-size selection, we have to determine the maximum step-size $\overline{\alpha}$ to assure the feasibility. This is computed by the formula $\overline{\alpha} = \min(\overline{\alpha}_1, \overline{\alpha}_2, \overline{\alpha}_3, \overline{\alpha}_4, \overline{\Delta}/\|d\|)$, where $\overline{\Delta}$ is an upper bound for the norm $\|x_{k+1} - x_k\|$ (parameter XMAX in our subroutines) and

$$\overline{\alpha}_1 = \min_{I_i^x=1, d_i<0} \frac{x_i^l - x_i}{d_i}, \qquad \overline{\alpha}_2 = \min_{I_i^x=2, d_i>0} \frac{x_i^u - x_i}{d_i},$$

$$\overline{\alpha}_3 = \min_{I_i^x=3, d_i<0} \frac{x_i^l - x_i}{d_i}, \qquad \overline{\alpha}_4 = \min_{I_i^x=3, d_i>0} \frac{x_i^u - x_i}{d_i}$$

(we use the value ∞ if a corresponding set is empty).

3. Inexact discrete Newton methods for sparse problems

Consider a general twice continuously differentiable function $F : R^n \to R$, where n is large, and assume that the Hessian matrix $G(x) = [G_{ij}(x)] = [\partial^2 F(x)/(\partial x_i \partial x_j)]$ is sparse. In this case, discrete versions of the Newton method can be efficiently used for seeking a local minimum of F. These methods are based on the fact that sufficiently sparse Hessian matrices can be estimated by using a small number of gradient differences [5]. We use the algorithm proposed in [35] in subroutines PNED, PNEC. The sparsity pattern of the Hessian matrix (only the upper part) is stored in the coordinate form (if ISPAS = 1) or in the standard compressed row format (if ISPAS = 2) using arrays IH and JH. For example, if the Hessian matrix has the pattern

$$G = \begin{bmatrix} * & * & * & 0 & * \\ * & * & 0 & * & 0 \\ * & 0 & * & 0 & * \\ 0 & * & 0 & * & 0 \\ * & 0 & * & 0 & * \end{bmatrix}$$

(asterisks denote nonzero elements), then arrays IH and JH contain elements

$$\mathtt{IH} = \begin{bmatrix} 1 & 1 & 1 & 1 & 2 & 2 & 3 & 3 & 4 & 5 \end{bmatrix}, \qquad \mathtt{JH} = \begin{bmatrix} 1 & 2 & 3 & 5 & 2 & 4 & 3 & 5 & 4 & 5 \end{bmatrix}$$

if ISPAS = 1 or

$$\mathtt{IH} = \begin{bmatrix} 1 & 5 & 7 & 9 & 10 & 11 \end{bmatrix}, \qquad \mathtt{JH} = \begin{bmatrix} 1 & 2 & 3 & 5 & 2 & 4 & 3 & 5 & 4 & 5 \end{bmatrix}$$

if ISPAS = 2. In the first case, nonzero elements in the upper part of the Hessian matrix can be sorted in an arbitrary order (not only by rows as in the above example) and arrays IH and JH have to be declared with lengths $n+m$, where m is the number of nonzero elements. In the second case, nonzero elements can be sorted only by rows. Components of IH contain addresses of the diagonal elements in this sequence and components of JH contain corresponding column indices (note that IH has $n+1$ elements and the last element is equal to $m+1$). Arrays IH and JH have to be declared with lengths $n+1$ and m, respectively.

Since the Hessian matrix can be indefinite, discrete versions of the Newton method are realized in the trust-region framework. Let B_k be a gradient-difference approximation of the Hessian matrix $G_k = G(x_k)$. Denote

$$Q_k(d) = \frac{1}{2} d^T B_k d + g_k^T d$$

the quadratic function which locally approximates the difference $F(x_k + d) - F(x_k)$,

$$\omega_k(d) = (B_k d + g_k)/\|g_k\|$$

the accuracy of the direction determination and

$$\rho_k(d) = \frac{F(x_k + d) - F(x_k)}{Q_k(d)}$$

the ratio of the actual and the predicted decrease of the objective function. Trust-region methods (used in subroutines PNED, PNEC, PGAD, PGAC) generate points $x_k \in R^n$, $k \in N$, in such a way that x_1 is arbitrary and

$$x_{k+1} = x_k + \alpha_k d_k, \quad k \in N, \tag{6}$$

where $d_k \in R^n$ are direction vectors and $\alpha_k \geq 0$ are step-sizes. Direction vectors $d_k \in R^n$ are chosen to satisfy conditions

$$\begin{aligned} \|d_k\| &\leq \Delta_k, &(7)\\ \|d_k\| &< \Delta_k \quad \Rightarrow \quad \|\omega_k(d_k)\| \leq \overline{\omega}_k, &(8)\\ -Q_k(d_k) &\geq \underline{\sigma}\|g_k\| \min(\|d_k\|, \|g_k\|/\|B_k\|), &(9) \end{aligned}$$

where $0 \leq \overline{\omega}_k \leq \overline{\omega} < 1$ and $0 < \underline{\sigma} < 1$ (we use the value $\overline{\omega} = 0.9$ in our subroutines; $\underline{\sigma}$ is a theoretical value given implicitly). Step-sizes $\alpha_k \geq 0$ are selected so that

$$\begin{aligned} \rho_k(d_k) \leq 0 \quad &\Rightarrow \quad \alpha_k = 0, &(10)\\ \rho_k(d_k) > 0 \quad &\Rightarrow \quad \alpha_k = 1. &(11) \end{aligned}$$

Trust-region radii $0 < \Delta_k \leq \overline{\Delta}$ are chosen in such a way that $0 < \Delta_1 \leq \overline{\Delta}$ (Δ_1 and $\overline{\Delta}$ are given by parameters XDEL and XMAX in our subroutines) and

$$\begin{aligned} \rho_k(d_k) < \underline{\rho} \quad &\Rightarrow \quad \underline{\beta}\|d_k\| \leq \Delta_{k+1} \leq \overline{\beta}\|d_k\|, &(12)\\ \underline{\rho} \leq \rho_k(d_k) \leq \overline{\rho} \quad &\Rightarrow \quad \Delta_{k+1} = \Delta_k, &(13)\\ \overline{\rho} < \rho_k(d_k) \quad &\Rightarrow \quad \Delta_{k+1} = \min(\gamma\Delta_{k+1}, \overline{\Delta}), &(14) \end{aligned}$$

where $0 < \underline{\beta} \leq \overline{\beta} < 1 < \gamma$ and $0 < \underline{\rho} < \overline{\rho} < 1$ (we use the values $\underline{\beta} = 0.05$, $\overline{\beta} = 0.75$, $\gamma = 2$, $\underline{\rho} = 0.1$, $\overline{\rho} = 0.9$ in our subroutines). Note that the initial trust-region radius Δ_1 is computed by a simple formula when XDEL $= 0$. This formula depends on a gradient norm $\|g_k\|$ and contains a lower bound for F (parameter FMIN in our subroutines) if INITS $= 1$. If INITS $= 0$, parameter FMIN need not be defined.

The direction vector satisfying (7)–(9) can be computed by four different strategies. Subroutines PNED,PGAD are based on matrix decomposition methods. They use either the Dennis–Mei (double dog-leg) method [9] (if MOS $=$ 1) or the Moré–Sorensen method [25] (if MOS $=$ 2). Subroutines PNEC,PGAC are based on matrix iterative methods. They use either the Steihaug–Toint method [30], [31] (if MOS1 $=$ 1) or the shifted Steihaug–Toint method [16] (if MOS1 $=$ 2). To simplify the description of these methods, we omit the outer index k and denote the inner index by i.

3.1 Matrix decomposition Moré–Sorensen trust region method

The most sophisticated methods are based on a computation of the optimal locally constrained step. A vector $d \in R^n$ is obtained by solving a subproblem

$$\text{minimize} \quad Q(d) = \frac{1}{2} d^T B d + g^T d \quad \text{subject to} \quad \|d\| \leq \Delta. \tag{15}$$

Necessary and sufficient conditions for this solution are

$$\|d\| \le \Delta, \quad (B+\lambda I)d + g = 0, \quad B + \lambda I \succeq 0, \quad \lambda \ge 0, \quad \lambda(\Delta - \|d\|) = 0 \tag{16}$$

(we use the symbol $\succeq$ for ordering by positive semidefiniteness). The Moré–Sorensen method [25] is based on solving a nonlinear equation

$$1/\|d(\lambda)\| = 1/\Delta \quad \text{with} \quad (B+\lambda I)d(\lambda) = -g$$

by the Newton method using a modification of the sparse Choleski decomposition of $B+\lambda I$ (we use the Gill–Murray decomposition [10]. More precisely, we determine $\underline{\mu}_1$ as the maximal diagonal element of the matrix $-B$, set $\underline{\lambda}_1 = \max(\underline{\mu}_1, 0)$, $\overline{\lambda}_1 = \|g\|/\Delta + \|B\|$, $\lambda_1 = \underline{\lambda}_1$ and for $i = 1, 2, 3, \dots$ we proceed in the following way. Carry out the Gill–Murray decomposition $B + \lambda_i I + E_i = R_i^T R_i$ (see [10]) for more details). If $E_i \neq 0$, determine a vector $v_i \in R^n$ such that $\|v_i\| = 1$ and $v_i^T(B+\lambda_i I)v_i < 0$, set $\underline{\mu}_i = \lambda_i - v_i^T(B+\lambda_i I)v_i$, $\underline{\lambda}_i = \underline{\mu}_i$, $\lambda_i = \underline{\lambda}_i$ and repeat this process (i.e., carry out the new Gill–Murray decomposition $B + \lambda_i I + E_i = R_i^T R_i$). If $E_i = 0$, compute a vector $d_i \in R^n$ by solving the equation $R_i^T R_i d_i + g = 0$. If $\|d_i\| > \overline{\delta}\Delta$, set $\underline{\lambda}_{i+1} = \lambda_i$ and $\overline{\lambda}_{i+1} = \overline{\lambda}_i$. If $\underline{\delta}\Delta \le \|d_i\| \le \overline{\delta}\Delta$ or $\|d_i\| < \underline{\delta}\Delta$ and $\lambda_i = 0$, set $d = d_i$ and terminate the computation. If $\|d_i\| < \underline{\delta}\Delta$ and $\lambda_i \neq 0$, set $\underline{\lambda}_{i+1} = \underline{\lambda}_i$, $\overline{\lambda}_{i+1} = \lambda_i$, determine a vector $v_i \in R^n$ such that $\|v_i\| = 1$ and $v_i^T d_i \ge 0$, which is a good approximation of an eigenvector of the matrix B corresponding to its minimal eigenvalue, and compute a number $\alpha_i \ge 0$ such that $\|d_i + \alpha_i v_i\| = \Delta$. If

$$\alpha_i^2 \|R_i v_i\|^2 \le (1 - \underline{\delta}^2)(\|R_i d_i\|^2 + \lambda_i \Delta^2),$$

set $d = d_i + \alpha_i v_i$ and terminate the computation, otherwise set $\underline{\mu}_i = \lambda_i - \|R_i v_i\|^2$. In this case or if $\|d_i\| > \overline{\delta}\Delta$, compute a vector $v_i \in R^n$ by solving the equation $R_i^T v_i = d_i$ and set

$$\lambda_{i+1} := \lambda_i + \frac{\|d_i\|^2}{\|v_i\|^2} \left(\frac{\|d_i\| - \Delta}{\Delta} \right).$$

If $\lambda_{i+1} < \underline{\lambda}_{i+1}$, set $\lambda_{i+1} = \underline{\lambda}_{i+1}$. If $\lambda_{i+1} > \overline{\lambda}_{i+1}$, set $\lambda_{i+1} = \overline{\lambda}_{i+1}$. We use the values $\underline{\delta} = 0.9$ and $\overline{\delta} = 1.1$ in our subroutines.

3.2 Matrix decomposition Dennis–Mei trust region method

The Moré–Sorensen method is very robust but requires 2-3 Choleski-type decompositions per iteration on average. Simpler methods are based on minimization of $Q(d)$ on a two-dimensional subspace containing the Cauchy step $d_C = -(g^T g / g^T B g)g$ and the Newton step $d_N = -B^{-1}g$. The Dennis–Mei double dog-leg method described in [9] seeks d as a linear combination of these two steps. This method uses vectors $d = d_N$ if $\|d_N\| \le \Delta$ or $d = (\Delta/\|d_C\|)d_C$ if $\|d_C\| \ge \Delta$. In the remaining case (if $\|d_C\| < \Delta < \|d_N\|$), d is a convex combination of d_C and τd_N, where $\tau = \max(d_C^T d_C / d_C^T d_N, \Delta/\|d_N\|)$, such that $\|d\| = \Delta$.

The Newton step is computed by using the sparse Gill–Murray decomposition [10], which has the form $B + E = LDL^T = R^T R$, where E is a positive semidefinite diagonal matrix (equal to zero when B is positive definite), L is a lower triangular matrix, D is a positive definite diagonal matrix and R is an upper triangular matrix. The matrix $LDL^T = R^T R$ then replaces B in $Q(d)$. The Dennis–Mei method requires only one Choleski-type decomposition per iteration.

3.3 Matrix iterative Steihaug–Toint trust region method

If B is not sufficiently sparse, then the sparse Choleski-type decomposition of B is expensive. In this case, methods based on preconditioned conjugate gradient (CG) iterations are more suitable. Steihaug [30] and Toint [31] proposed a method based on the fact that $Q(d_{i+1}) < Q(d_i)$ and $\|d_{i+1}\|_C > \|d_i\|_C$ (where $\|d_i\|_C^2 = d_i^T C d_i$) hold in the preconditioned CG iterations if CG coefficients are positive. We either obtain an unconstrained solution with a sufficient precision or stop on the trust-region boundary if a negative curvature is indicated or if the trust-region is left. More precisely, we set $d_1 = 0$, $g_1 = g$, $p_1 = -C^{-1}g$ and for $i = 1, 2, 3, \dots$ we proceed in the following way. If $\|g_i\| \le \overline{\omega}\|g\|$, then set $d = d_i$ and terminate the computation, otherwise set

$$q_i = Bp_i, \qquad \alpha_i = g_i^T C^{-1} g_i / p_i^T q_i.$$

If $\alpha_i \le 0$, determine $\alpha_i \ge 0$ in such a way that $\|d_i + \alpha_i p_i\| = \Delta$, set $d := d_i + \alpha_i p_i$ and terminate the computation, otherwise compute $d_{i+1} = d_i + \alpha_i p_i$. If $\|d_{i+1}\| \ge \Delta$, determine $\alpha_i \ge 0$ in such a way that $\|d_i + \alpha_i p_i\| = \Delta$, set $d := d_i + \alpha_i p_i$ and terminate the computation, otherwise compute

$$\begin{aligned} g_{i+1} &= g_i + \alpha_i q_i, \qquad \beta_i = g_{i+1}^T C^{-1} g_{i+1} / g_i^T C^{-1} g_i \\ p_{i+1} &= -C^{-1} g_{i+1} + \beta_i p_i. \end{aligned}$$

The matrix C serves as a preconditioner (symmetric and positive definite). If `MOS2` $= 1$ (`MOS2` is a parameter of subroutines `PNEC,PGAC`), then no preconditioning is performed ($C = I$), if `MOS2` $= 2$, an incomplete Choleski decomposition of the matrix B is used, and if `MOS2` $= 3$, a preliminary solution obtained by the incomplete Choleski decomposition can be accepted. In this case, we first compute $p_1 = -C^{-1}g$. If $\|Bp_1 + g\| \le \overline{\omega}\|g\|$, we set $d = p_1$ and terminate the computation, otherwise we continue by CG iterations as above.

There are two possible ways that the Steihaug–Toint method can be preconditioned. The first way uses norms $\|d_i\|_{C_i}$ (instead of $\|d_i\|$) in (7)–(14), where C_i are preconditioners chosen. This possibility has been tested in [11] and showed that such a way is not always efficient. This is caused by the fact that norms $\|d_i\|_{C_i}$, $i \in N$, vary considerably in the major iterations and preconditioners C_i, $i \in N$, can be ill-conditioned. The second way uses Euclidean norms in (7)–(14) even if arbitrary preconditioners C_i, $i \in N$, are used. In this case the trust region can be left prematurely and the direction vector obtained can be farther from the optimal locally-constrained step than that obtained without preconditioning. This shortcoming is usually compensated by the rapid convergence of the preconditioned CG method. Our computational experiments indicated that the second way is more efficient in general and we use it in our subroutines.

3.4 Matrix iterative shifted Steihaug–Toint trust region method

Since the optimal locally constrained step has to satisfy the equation $(B+\lambda I)d+g = 0$, where $\lambda \ge 0$ is the optimal Lagrange multiplier (see (16)), it seems to be advantageous to apply the Steihaug–Toint method to the subproblem

$$\text{minimize} \quad \tilde{Q}(d) = Q_{\tilde{\lambda}}(d) = \frac{1}{2} d^T (B + \tilde{\lambda} I) d + g^T d \quad \text{s.t.} \quad \|d\| \le \Delta, \tag{17}$$

where $\tilde{\lambda} \ge 0$ is an approximation of the optimal λ. The number $\tilde{\lambda} \ge 0$ is found by solving a small-size subproblem

$$\text{minimize} \quad \frac{1}{2} \tilde{d}^T T \tilde{d} + \|g\| e_1^T \tilde{d} \quad \text{s.t.} \quad \|\tilde{d}\| \le \Delta \tag{18}$$

with the tridiagonal matrix T obtained by using a small number of Lanczos steps. This method combines good properties of the Moré–Sorensen and the Steihaug–Toint methods and can be successfully preconditioned by the second way described in the previous subsection. The point on the trust-region boundary obtained by this method is usually closer to the optimal solution in comparison with the point obtained by the original Steihaug–Toint method.

The above considerations form a basis for the shifted Steihaug–Toint method proposed in [16]. This method consists of the three steps:

(1) Let $m =$ `MOS1` (the default value is `MOS1` $= 5$). Determine a tridiagonal matrix T of order m by using m steps of the (unpreconditioned) Lanczos method (described, e.g., in [11], [14]) applied to the matrix B with the initial vector g.

(2) Solve the subproblem (18) by using the Moré–Sorensen method described in Section 3.1 to obtain a Lagrange multiplier $\tilde{\lambda}$.

(3) Apply the (preconditioned) Steihaug–Toint method described in Section 3.3 to the subproblem (17) to obtain a direction vector $d = d(\tilde{\lambda})$.

Let $\tilde{\lambda}$ be the Lagrange multiplier of small-size subproblem (18) and λ be the Lagrange multiplier obtained by the Moré–Sorensen method applied to the original trust-region subproblem (15). It can be shown (see [16]) that $0 \le \tilde{\lambda} \le \lambda$. This inequality assures that $\lambda = 0$ implies $\tilde{\lambda} = 0$ so $\|d\| < \Delta$ implies $\tilde{\lambda} = 0$. Thus the shifted Steihaug–Toint method reduces to the standard one in this case. At the same time, if B is positive definite and $\tilde{\lambda} > 0$, then one has $\Delta \le \|(B + \tilde{\lambda} I)^{-1} g\| < \|B^{-1} g\|$. Thus the unconstrained minimizer of the shifted quadratic function (17) is closer to the trust-region boundary than the unconstrained minimizer of the original quadratic function (15) and we can expect that $d(\tilde{\lambda})$ is closer to the optimal locally constrained step than $d(0)$. Finally, if $\tilde{\lambda} > 0$, then the matrix $B + \tilde{\lambda} I$ is better conditioned than B and we can expect that the shifted Steihaug–Toint method will converge more rapidly than the original one.

4. Methods for partially separable problems

Consider a function of the form

$$F(x) = \sum_{i=1}^{n_a} f_i(x), \tag{19}$$

where $f_i(x)$, $1 \le i \le n_a$ (n_a is usually large), are smooth particular functions depending on a small number of variables (n_i, say). In this case, the Jacobian matrix $J(x) = [J_{ij}(x)] = [\partial f_i(x)/\partial x_j]$ is sparse. In subroutines `PSED`, `PSEC`, and `PSEN`, the sparsity pattern of the Jacobian matrix is stored in the coordinate form (if `ISPAS` = `1`) or in the standard compressed row format (if `ISPAS` = `2`) using arrays `IAG` and `JAG`. For example, if the Jacobian matrix has the pattern

$$J = \begin{bmatrix} * & * & 0 & * \\ * & * & * & 0 \\ * & 0 & 0 & * \\ 0 & * & * & 0 \\ * & 0 & * & 0 \end{bmatrix}$$

(asterisks denote nonzero elements) then arrays `IAG` and `JAG` contain elements

$$\texttt{IAG} = \begin{bmatrix} 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 4 & 4 & 5 & 5 \end{bmatrix}, \quad \texttt{JAG} = \begin{bmatrix} 1 & 2 & 4 & 1 & 2 & 3 & 1 & 4 & 2 & 3 & 1 & 3 \end{bmatrix},$$

if `ISPAS` = `1` or

$$\texttt{IAG} = \begin{bmatrix} 1 & 4 & 7 & 9 & 11 & 13 \end{bmatrix}, \qquad \texttt{JAG} = \begin{bmatrix} 1 & 2 & 4 & 1 & 2 & 3 & 1 & 4 & 2 & 3 & 1 & 3 \end{bmatrix},$$

if `ISPAS` = `2`. In the first case, nonzero elements can be sorted in an arbitrary order (not only by rows as in the above example). Arrays `IAG` and `JAG` have to be declared with lengths $n_a + m_a$ and m_a, respectively, where m_a is the number of nonzero elements. In the second case, nonzero elements can be sorted only by rows. Components of `IAG` contain total numbers of nonzero elements in all previous rows increased by 1 and elements of `JAG` contain corresponding column indices (note that `IAG` has $n_a + 1$ elements and the last element is equal to $m_a + 1$). Arrays `IAG` and `JAG` have to be declared with lengths $n_a + 1$ and m_a, respectively. This representation of sparse Jacobian matrices is also used in subroutines `PGAD`, `PGAC`, `PMAX`, `PSUM`, `PEQN`, `PEQL` described in the subsequent sections.

Using the sparsity pattern of the Jacobian matrix, we can define packed gradients $\hat{g}_i(x) \in R^{n_i}$ and packed Hessian matrices $\hat{G}_i(x) \in R^{n_i \times n_i}$ of functions $f_i(x)$, $1 \le i \le n_a$, as dense but small-size vectors and matrices. Note that $\hat{g}_i(x) = Z_i^T g_i(x)$, $\hat{G}_i(x) = Z_i^T G_i(x) Z_i$ and $g_i(x) = Z_i \hat{g}_i(x)$, $G_i(x) = Z_i \hat{G}_i(x) Z_i^T$, $1 \le i \le n_a$, where $g_i(x)$ and $G_i(x)$ are original gradients and Hessian matrices of functions $f_i(x)$, respectively, and $Z_i \in R^{n \times n_i}$ are matrices containing columns of the unit matrix corresponding to the variables appearing in $f_i(x)$.

Methods for partially separable problems are implemented in the line-search framework mentioned in Section 2. A direction vector d_k is computed by solving a system of linear equations with a matrix B_k, which is an approximation of the Hessian matrix computed from approximations of packed Hessian matrices, see (22) below. Subroutines `PSED` and `PSEN` use the Gill–Murray matrix decomposition. In this case, B_k is replaced by $L_k D_k L_k^T = B_k + E_k$, where L_k is a lower triangular matrix, D_k is a positive definite diagonal matrix, and E_k is a positive semidefinite diagonal matrix chosen in such a way that $B_k + E_k$ is positive definite (more details are given in [10]). Subroutine `PSEC` uses the preconditioned conjugate gradient method described in Subsection 2.1, where multiplications by B_k are explicitly used instead of (4) and $\overline{\omega} = 0.9$ in (5).

4.1 Partitioned variable metric methods

Subroutines `PSED`, `PSEC` are based on partitioned variable metric updates [12], which consider each particular function separately. Thus approximations $\hat{B}_i$, $1 \le i \le n_a$, of the packed Hessian matrices $\hat{G}_i(x)$ are updated by using the quasi-Newton conditions $\hat{B}_i^+ \hat{s}_i = \hat{y}_i$, where $\hat{s}_i = Z_i^T s = Z_i^T (x^+ - x)$ and $\hat{y}_i = \hat{g}_i^+ - \hat{g}_i = Z_i^T (g_i^+ - g_i)$ (we omit outer index k and replace index $k+1$ by $+$ in this section). Therefore, a variable metric update can be used for each of the particular functions. However, there is a difference between the classic and the partitioned approach, since conditions $\hat{s}_i^T \hat{y}_i > 0$, which are necessary for positive definiteness of $\hat{B}_i^+$, are not guaranteed for all $1 \le i \le n_a$. This difficulty is unavoidable and an efficient algorithm has to handle this situation.

Subroutines `PSED`, `PSEC` use three strategies. If `MET` = `1`, then the safeguarded partitioned BFGS updates

$$\begin{aligned} \hat{B}_i^+ &= \hat{B}_i + \frac{\hat{y}_i \hat{y}_i^T}{\hat{s}_i^T \hat{y}_i} - \frac{\hat{B}_i \hat{s}_i (\hat{B}_i \hat{s}_i)^T}{\hat{s}_i^T \hat{B}_i \hat{s}_i}, \quad && \hat{s}_i^T \hat{y}_i > 0, \\ \hat{B}_i^+ &= \hat{B}_i, && \hat{s}_i^T \hat{y}_i \le 0 \end{aligned} \tag{20}$$

are used. If `MET` = `2`, then the BFGS updates are combined with the rank-one updates

$$\begin{aligned} \hat{B}_i^+ &= \hat{B}_i + \frac{(\hat{y}_i - \hat{B}_i \hat{s}_i)(\hat{y}_i - \hat{B}_i \hat{s}_i)^T}{\hat{s}_i^T (\hat{y}_i - \hat{B}_i \hat{s}_i)}, \quad && |\hat{s}_i^T (\hat{y}_i - \hat{B}_i \hat{s}_i)| \ge \varepsilon_M |\hat{s}_i^T \hat{B}_i \hat{s}_i|, \\ \hat{B}_i^+ &= \hat{B}_i, && |\hat{s}_i^T (\hat{y}_i - \hat{B}_i \hat{s}_i)| < \varepsilon_M |\hat{s}_i^T \hat{B}_i \hat{s}_i|, \end{aligned} \tag{21}$$

where ε_M is the machine precision. We use a strategy, which is based on the observation that (20) usually leads to the loss of convergence if too many particular functions have indefinite Hessian matrices. We start with the partitioned BFGS update (20). If $n_- \ge \theta n_a$, where n_- is a number of particular functions with a negative curvature and θ is a threshold value, then (21) is used for all particular functions in all subsequent iterations (we use the value $\theta = 1/2$ in the subroutines `PSED`, `PSEC`). If `MET` = `3`, then packed matrices $\hat{B}_i \approx \hat{G}_i(x)$ are computed by using gradient differences. This strategy is in fact the partitioned discrete Newton method.

A disadvantage of partitioned variable metric methods (`MET` = `1`, `MET` = `2`) is the fact that approximations of packed Hessian matrices need to be stored. Therefore, the number of stored elements can be much greater than the number of nonzero elements in the sparse Hessian matrix. Moreover, operations with packed Hessian matrices (decomposition, multiplication) are usually unsuitable (time consuming). Thus the sparse approximation of the Hessian matrix

$$B = \sum_{i=1}^{n_a} Z_i \hat{B}_i Z_i^T \tag{22}$$

(stored as in Section 3) is constructed in subroutines `PSED`, `PSEC`. Then a direction vector d, used in line search, is computed by solving the linear system $Bd = -g$. The partitioned Newton method (`MET` = `3`) does not store packed matrices $\hat{B}_i$, $1 \le i \le n_a$, since they are immediately added to matrix (22).

4.2 Partitioned variable metric method for nonsmooth functions

Assume that functions $f_i(x)$, $1 \le i \le n_a$, appearing in (19), are nonsmooth, locally Lipschitz, and we are able to compute (Clarke) subgradients $g_i \in \partial f_i(x)$, $1 \le i \le n_a$, at any point $x \in R^n$. Then also $F(x)$ is locally Lipschitz and since locally Lipschitz function is differentiable almost everywhere by the Rademacher theorem, usually $\partial F(x) = \{\nabla F(x)\}$. A special feature of nonsmooth functions is the fact that the gradient $\nabla F(x)$ changes discontinuously and is not small in the neighborhood of a local minimum. Thus the standard optimization methods cannot be used efficiently.

The most commonly used approach for solving nonsmooth optimization problems is based on the bundle principle. In this case, values $F(x_k)$, $g(x_k) \in \partial F(x_k)$ at a single point x_k are replaced by a bundle of values $F_j = F(z_j)$, $g_j \in \partial F(z_j)$ obtained at trial points z_j, $j \in \mathcal{J}_k \subset \{1, \dots, k\}$. This bundle of values serves for defining a piecewise quadratic function (with a quadratic regularizing term), which is used for direction determination by solving a quadratic programming subproblem. Usually, the bundle contains many dense subgradients. Thus a dense quadratic programming subproblem with many constraints has to be solved, which is unsuitable in the large-scale cases. This disadvantage can be overcome by the bundle variable metric method described in [36], which uses a bundle with three subgradients at most for defining a quadratic programming subproblem. Subroutine `PSEN` is based on the partitioned bundle variable metric method described in [23], which combines ideas from [36] with partitioned variable metric updates.

Using aggregate subgradients $\tilde{g}_k$ and aggregated subgradient locality measures $\tilde{\beta}_k$ (where $\tilde{g}_1 = g_1$ and $\tilde{\beta}_1 = 0$ in the first iteration), the partitioned bundle variable metric method generates a sequence of basic points $\{x_k\} \subset R^n$ and a sequence of trial points $\{z_k\} \subset R^n$ such that

$$x_{k+1} = x_k + \alpha_k^L d_k, \quad z_{k+1} = x_k + \alpha_k^R d_k,$$

where $d_k = -(B_k^{-1} + \rho I)\tilde{g}_k$ is a direction vector and $\alpha_k^R > 0$, $\alpha_k^R \ge \alpha_k^L \ge 0$ are appropriately chosen step-sizes. At the same time, B_k is a matrix obtained by partitioned variable metric updates and ρ is a correction parameter (parameter `ETA3` in subroutine `PSEN`). Step-sizes α_k^R and α_k^L are chosen by a special line-search procedure in such a way that either

$$F(x_k + \alpha_k^L d_k) \le F(x_k) - \varepsilon_L \alpha_L^R w_k$$

or

$$d_k^T g(x_k + \alpha_k^R d_k) \ge \gamma_{k+1} - \varepsilon_R w_k.$$

Here $0 < \varepsilon_L < 1/2$, $\varepsilon_L < \varepsilon_R < 1$ are suitable constants (we use the values $\varepsilon_L = 10^{-4}$, $\varepsilon_R = 0.25$ in our subroutine), $g(x_k + \alpha_k^R d_k) \in \partial F(x_k + \alpha_k^R d_k)$, $w_k = -(\tilde{g}_k)^T d_k + 2\tilde{\beta}_k$ and

$$\gamma_{k+1} = \max\left(|F(x^k) - F(x_k + \alpha_k^R d_k) + \alpha_k^R d_k^T g(x_k + \alpha_k^R d_k)|, \gamma \|\alpha_k^R d_k\|^2\right)$$

where γ is a subgradient locality measure parameter (parameter `ETA5` in subroutine `PSEN`). In the first case (descent step) we set $z_{k+1} = x_{k+1} = x_k + \alpha_k^L d_k$, $\beta_{k+1} = 0$ and substitute $\tilde{\beta}_{k+1} = 0$, $\tilde{g}_{k+1} = g_{k+1} \in \partial F(z_{k+1})$. In the second case (null step) we set $z_{k+1} = x_k + \alpha_k^R d_k$, $x_{k+1} = x_k$, $\beta_{k+1} = \gamma_{k+1}$ and determine $\tilde{\beta}_{k+1}$, $\tilde{g}_{k+1}$ by the aggregation procedure.

The aggregation procedure is very simple. Denoting by l the lowest index satisfying $x_l = x_k$ (index of the iteration after the last descent step) and using the subgradients g_l, g_{k+1}, $\tilde{g}_k$ and the subgradient locality measures $\beta_l = 0$, β_{k+1}, $\tilde{\beta}_k$, we determine multipliers

$$\lambda_k^1 \ge 0, \quad \lambda_k^2 \ge 0, \quad \lambda_k^3 \ge 0, \quad \lambda_k^1 + \lambda_k^2 + \lambda_k^3 = 1,$$

which minimize the quadratic function

$$(\lambda^1 g_l + \lambda^2 g_{k+1} + \lambda^3 \tilde{g}_k)^T (B_k^{-1} + \rho I)(\lambda^1 g_l + \lambda^2 g_{k+1} + \lambda^3 \tilde{g}_k) + 2(\lambda^1 \beta_l + \lambda^2 \beta_{k+1} + \lambda^3 \tilde{\beta}_k),$$

and set

$$\tilde{g}_{k+1} = \lambda_k^1 g_l + \lambda_k^2 g_{k+1} + \lambda_k^3 \tilde{g}_k, \quad \tilde{\beta}_{k+1} = \lambda_k^1 \beta_l + \lambda_k^2 \beta_{k+1} + \lambda_k^3 \tilde{\beta}_k.$$

After obtaining points x_{k+1}, z_{k+1} and subgradient $g_{k+1} \in \partial F(z_{k+1})$, the matrix B_k is updated. To satisfy conditions for the global convergence, we use special symmetric rank-one updates after null steps. This guarantees that elements of B_k do not decrease. After descent steps, the safeguarded BFGS updates can be used. As the function $F(x)$ is partially separable, the matrix B_k can be expressed in form (22) (we omit outer index k and replace index $k+1$ by $+$). Thus we set

$$\begin{aligned}
\hat{B}_i^+ &= \hat{B}_i + \frac{\hat{y}_i \hat{y}_i^T}{\hat{s}_i^T \hat{y}_i} - \frac{\hat{B}_i \hat{s}_i (\hat{B}_i \hat{s}_i)^T}{\hat{s}_i^T \hat{B}_i \hat{s}_i}, \quad && \hat{s}_i^T \hat{y}_i > 0, \\
\hat{B}_i^+ &= \hat{B}_i, && \hat{s}_i^T \hat{y}_i \le 0
\end{aligned}$$

after a descent step or

$$\begin{aligned}
\hat{B}_i^+ &= \hat{B}_i + \frac{(\hat{y}_i - \hat{B}_i \hat{s}_i)(\hat{y}_i - \hat{B}_i \hat{s}_i)^T}{\hat{s}_i^T (\hat{y}_i - \hat{B}_i \hat{s}_i)}, \quad && \hat{s}_i^T (\hat{y}_i - \hat{B}_i \hat{s}_i) \ge \varepsilon_M \hat{s}_i^T \hat{B}_i \hat{s}_i, \\
\hat{B}_i^+ &= \hat{B}_i, && \hat{s}_i^T (\hat{y}_i - \hat{B}_i \hat{s}_i) < \varepsilon_M \hat{s}_i^T \hat{B}_i \hat{s}_i
\end{aligned}$$

after a null step (ε_M is the machine precision). Here $\hat{s}_i = Z_i^T s_i$ and $\hat{y}_i = \hat{g}_i^+ - \hat{g}_i = Z_i^T (g_i^+ - g_i)$, where g_i^+ and g_i are subgradients computed at points z^+ and x, respectively (aggregated subgradients $\tilde{g}_i^+$ and $\tilde{g}_i$ are not used in variable metric updates).

Since the quadratic programming subproblem used in the aggregation procedure is very simple, initial step-size $\alpha = 1$ need not be a good choice in connection with its solution. Therefore we store and use a bundle of values $F_j = F(z_j)$, $g_j \in \partial F(z_j)$ obtained at trial points z_j, $j \in \mathcal{J}_k = \{k-m, \dots, k\}$. Here m is a size of the bundle (parameter `MB` of subroutine `PSEN`). These values are used for the construction of a piecewise linear function which serves for determination of a better initial step-size. More details are given in [23].

5. Hybrid methods for nonlinear least-squares

Consider a function of the form

$$F(x) = \frac{1}{2} \sum_{i=1}^{n_a} f_i^2(x) = \frac{1}{2} f^T(x) f(x) \tag{23}$$

(sum of squares), where $f_i(x)$, $1 \le i \le n_a$ (n_a is usually large), are smooth functions depending on a small number of variables (n_i, say). In this case, the Jacobian matrix $J(x) = [J_{ij}(x)] = [\partial f_i(x)/\partial x_j]$ is sparse. The sparsity pattern of the Jacobian matrix is stored using arrays `IAG` and `JAG` in the way described in Section 4.

Using the Jacobian matrix, we can express the gradient $g(x)$ and the Hessian matrix $G(x)$ in the form

$$\begin{aligned}
g(x) &= \sum_{i=1}^{n_a} f_i(x) g_i(x) = J^T(x) f(x), \\
G(x) &= \sum_{i=1}^{n_a} \big(g_i(x) g_i^T(x) + f_i(x) G_i(x) \big) = J^T(x) J(x) + C(x)
\end{aligned}$$

($G_i(x)$ are Hessian matrices of $f_i(x)$, $1 \le i \le n_a$, and $C(s)$ is a second order term). The well-known Gauss-Newton method uses the matrix $J^T(x)J(x)$ instead of the Hessian matrix $G(x) = J^T(x)J(x) + C(x)$ (i.e., it omits the second order information contained in $C(x)$). We assume that the matrix $J^T(x)J(x)$ is sparse (then also $C(x)$ is sparse).

The matrix $J^T(x)J(x)$ is frequently ill-conditioned (even singular) so that the Gauss-Newton method and its modifications require trust-region realizations. For computing a trust-region step, subroutine `PGAD` uses matrix decomposition methods and subroutine `PGAC` uses matrix iterative methods. These methods and their choices (using variables `MOS` `MOS1`, and `MOS2`) are described in Section 3.

If the minimum value $F(x^*)$ is large (large residual problem), the Gauss-Newton method can be inefficient. Therefore, modifications that use the estimation of the second-order term have been developed. These modifications are based on the fact (proven in [1]) that $(F_k - F_{k+1})/F_k \to 1$ if $F_k \to 0$ Q-superlinearly and $(F_k - F_{k+1})/F_k \to 0$ if $F_k \to F^* > 0$. Thus we can use the following philosophy. Let x_{k+1} be a vector obtained by the trust-region strategy described in Section 3. If $x_{k+1} \neq x_k$, we compute $F_{k+1} = F(x_{k+1})$, $J_{k+1} = J(x_{k+1})$ and set

$$\begin{aligned} B_{k+1} &= J_{k+1}^T J_{k+1}, & F_k - F_{k+1} &> \underline{\vartheta} F_k, \\ B_{k+1} &= J_{k+1}^T J_{k+1} + C_{k+1}, & F_k - F_{k+1} &\le \underline{\vartheta} F_k, \end{aligned}$$

where C_{k+1} is an approximation of the second order term and $\underline{\vartheta}$ is a suitable value (parameter `ETA` in subroutines `PGAD` and `PGAC`).

Large-scale correction matrix C_{k+1} cannot be obtained by dense variable metric updates [1], which are frequently used for medium-size problems. Fortunately, simple corrections utilizing sparsity also increase the efficiency of the Gauss-Newton method. In subroutines `PGAD` and `PGAC` we have implemented three hybrid methods proposed in [15] (specified by the variable `MEC`), which are described in the subsequent subsections. To simplify the notation, we omit outer index k and replace index $k+1$ by $+$.

5.1 Gauss-Newton method with sparse variable metric corrections

If `MEC` = `1`, the sparse variable metric corrections (Marwil updates) are used. In the first iteration (or after a restart) we use the matrix $B = J^T J$. In the subsequent iterations we set

$$\begin{aligned} B^+ &= (J^+)^T J^+, & F - F^+ &> \underline{\vartheta} F, \\ B^+ &= \mathcal{P}_S \mathcal{P}_{QG}((J^+)^T J^+), & F - F^+ &\le \underline{\vartheta} F, \end{aligned}$$

where $\mathcal{P}_S$ realizes an orthogonal projection into the subspace of symmetric matrices of order n and $\mathcal{P}_{QG}$ realizes an orthogonal projection into the intersection of the subspace of matrices having the same sparsity pattern as $J^T J$ and the linear manifold of matrices satisfying the quasi-Newton condition $Ws = y$ with $s = x^+ - x$, $y = g^+ - g$. Thus

$$\mathcal{P}_S W = (W + W^T)/2,$$

$$\begin{aligned} (\mathcal{P}_G W)_{ij} &= W_{ij}, & (J^T J)_{ij} &\neq 0, \\ (\mathcal{P}_G W)_{ij} &= 0, & (J^T J)_{ij} &= 0, \end{aligned}$$

for a given square matrix W, and

$$\mathcal{P}_{QG}((J^+)^T J^+) = \mathcal{P}_G((J^+)^T J^+ + u s^T),$$

where $u \in R^n$ is a solution of the linear system $Du = y - (J^+)^T J^+ s$ with a diagonal matrix D such that

$$D_{ii} = \sum_{(J^T J)_{ij} \neq 0} (e_j^T s)^2$$

(e_j is the j-th column of the unit matrix).

5.2 Gauss-Newton method with the Newton corrections

If `MEC` = `2`, the Newton corrections are used. In the first iteration (or after a restart) we use the matrix $B = J^T J$. In the subsequent iterations we set

$$\begin{aligned} B^+ &= (J^+)^T J^+, & F - F^+ &> \underline{\vartheta} F, \\ B^+ &= (J^+)^T J^+ + \sum_{i=1}^{n_a} f_i^+ G_i^+, & F - F^+ &\le \underline{\vartheta} F, \end{aligned}$$

where G_i^+, $1 \le i \le n_a$, are approximations of Hessian matrices determined by using gradient differences at the point x^+.

5.3 Gauss-Newton method with partitioned variable metric corrections

If $\mathtt{MEC} = \mathtt{3}$, the partitioned variable metric corrections (symmetric rank-one updates) are used. In the first iteration (or after a restart) we use the matrix $B = J^T J$. In the subsequent iterations we set

$$\begin{aligned} B^+ &= (J^+)^T J^+, & F - F^+ > \underline{\vartheta} F, \\ B^+ &= (J^+)^T J^+ + \sum_{i=1}^{n_a} f_i^+ Z_i \hat{B}_i^+ Z_i^T, & F - F^+ \le \underline{\vartheta} F, \end{aligned}$$

where Z_i, $1 \le i \le n_a$, are matrices defined in Section 4 and $\hat{B}_i^+$, $1 \le i \le n_a$, are packed matrices updated using symmetric rank-one formulas

$$\begin{aligned} \hat{B}_i^+ &= \hat{B}_i + \frac{(\hat{y}_i - \hat{B}_i \hat{s}_i)(\hat{y}_i - \hat{B}_i \hat{s}_i)^T}{\hat{s}_i^T (\hat{y}_i - \hat{B}_i \hat{s}_i)}, & |\hat{s}_i^T (\hat{y}_i - \hat{B}_i \hat{s}_i)| \ge \varepsilon_M |\hat{s}_i^T \hat{B}_i \hat{s}_i|, \\ \hat{B}_i^+ &= \hat{B}_i, & |\hat{s}_i^T (\hat{y}_i - \hat{B}_i \hat{s}_i)| < \varepsilon_M |\hat{s}_i^T \hat{B}_i \hat{s}_i| \end{aligned}$$

(ε_M is the machine precision), where $\hat{B}_i = I$, $1 \le i \le n_a$, in the first iteration (or after a restart). A disadvantage of partitioned variable metric corrections is the fact that approximations of packed Hessian matrices need to be stored. Thus the choice $\mathtt{MEC} = \mathtt{3}$ is usually less suitable than choices $\mathtt{MEC} = \mathtt{1}$ and $\mathtt{MEC} = \mathtt{2}$.

6. Primal interior-point methods for minimax optimization

Consider a function of the form

$$F(x) = \max_{1 \le i \le n_a} f_i(x) \tag{24}$$

(pointwise maximum), where $f_i(x)$, $1 \le i \le n_a$ (n_a is usually large), are smooth functions depending on a small number of variables (n_i, say). In this case, the Jacobian matrix $J(x) = [J_{ij}(x)] = [\partial f_i(x)/\partial x_j]$ is sparse. The sparsity pattern of the Jacobian matrix is stored using arrays `IAG` and `JAG` in the way described in Section 4.

Primal interior point methods for minimax optimization, proposed in [17], are based on three basic ideas. First, minimization of F is equivalent to the nonlinear programming problem with $n+1$ variables $x \in R^n$, $z \in R$:

$$\text{minimize} \quad z \quad \text{subject to} \quad f_i(x) \le z, \quad 1 \le i \le n_a. \tag{25}$$

Secondly, this constrained problem is replaced by a sequence of unconstrained problems

$$\text{minimize} \quad B_\mu(x,z) = z - \mu \sum_{i=1}^{n_a} \log(z - f_i(x)),$$

where $z > F(x)$ and $\mu > 0$ (we assume that $\mu \to 0$ monotonically). Finally, the extra variable z is eliminated by solving the scalar nonlinear equation

$$\sum_{i=1}^{n_a} \frac{\mu}{z - f_i(x)} = 1, \tag{26}$$

which follows from the first-order necessary conditions for the barrier function $B_\mu(x,z)$ with fixed $x \in R^n$. The above equation has a unique root $z_\mu(x)$ such that $F(x) + \mu \le z_\mu(x) \le F(x) + n_a\mu$. This approach leads to inexact unconstrained minimizations of functions $B_\mu(x) = B_\mu(x, z_\mu(x))$ for suitable values of μ using the fact that

$$\nabla B_\mu(x) = g_\mu(x) = J^T(x) u_\mu(x)$$

and

$$\nabla^2 B_\mu(x) = G_\mu(x) + J^T(x) V_\mu(x) J(x) - \frac{J^T(x) V_\mu(x) e e^T V_\mu(x) J(x)}{e^T V_\mu(x) e},$$

where

$$u_\mu(x) = \begin{bmatrix} \mu/(z_\mu(x) - f_1(x)) \\ \dots \\ \mu/(z_\mu(x) - f_{n_a}(x)) \end{bmatrix}, \quad e = \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix},$$

$$G_\mu(x) = \sum_{i=1}^{n_a} e_i^T u_\mu(x) \nabla^2 f_i(x),$$

$$V_\mu(x) = \operatorname{diag}\left(\mu/(z_\mu(x) - f_1(x))^2, \dots, \mu/(z_\mu(x) - f_{n_a}(x))^2\right).$$

The Hessian matrix $\nabla^2 B_\mu(x)$ of the barrier function $B_\mu(x)$ is positive definite if $G_\mu(x)$ (the Hessian matrix of the Lagrangian function) is positive definite.

Subroutine `PMAX` is based on a line search realization of the Newton-like methods. Thus $x^+ = x + \alpha d$, where $\nabla^2 B_\mu(x) d = -g_\mu(x)$ and α is a suitable step-size. In fact, we use an approximation of $\nabla^2 B_\mu(x)$, such that $G_\mu(x)$ is determined either by partitioned variable metric updates described in Section 4 (if `MED` $= 1$) or by gradient differences as in [5] (if `MED` $= 2$). In the second case, the matrix $G_\mu(x)$ is not positive definite in general so a restart strategy guaranteeing descent is used (more details are given in [17]). If `MED` $= 1$, then we define reduced approximations of the Hessian matrices $\tilde{G}_i = Z_i^T G_i Z_i$,

$1 \le i \le n_a$, as in Section 4. New reduced approximations of the Hessian matrices are computed by the formulas

$$\begin{aligned}\tilde{G}_i^+ &= \frac{1}{\tilde{\gamma}_i}\left(\tilde{G}_i - \frac{\tilde{G}_i\tilde{s}_i\tilde{s}_i^T\tilde{G}_i}{\tilde{s}_i^T\tilde{G}_i\tilde{s}_i}\right) + \frac{\tilde{y}_i\tilde{y}_i^T}{\tilde{s}_i^T\tilde{y}_i}, \quad && \tilde{s}_i^T\tilde{y}_i > 0, \\ \tilde{G}_i^+ &= \tilde{G}_i, && \tilde{s}_i^T\tilde{y}_i \le 0,\end{aligned} \tag{27}$$

where

$$\tilde{s}_i = Z_i^T(x^+ - x), \quad \tilde{y}_i = Z_i^T(\nabla f_i(x^+) - \nabla f_i(x)), \quad 1 \le i \le n_a,$$

and where either $\tilde{\gamma}_i = 1$ or $\tilde{\gamma} = \tilde{s}_i^T\tilde{G}_i\tilde{s}_i/\tilde{s}_i^T\tilde{y}_i$. The particular choice of $\tilde{\gamma}_i$ is determined by the controlled scaling strategy described in [19]. In the first iteration we set $\tilde{G}_i = I$, $1 \le i \le n_a$, where I are unit matrices of suitable orders. Finally, $G_i^+ = Z_i\tilde{G}_i^+Z_i^T$, $1 \le i \le n_a$.

A very important part of the primal interior point method is an update of the barrier parameter μ. There are two requirements, which play opposite roles. First, $\mu \to 0$ should hold, since this is the main property of every interior-point method. On the other hand, round-off errors can cause that $z_\mu(x) = F(x)$ when μ is too small and $|F(x)|$ is sufficiently large (since $F(x)+\mu \le z_\mu(x) \le F(x)+n_a\mu$), which leads to a breakdown (division by $z_\mu(x) - F(x) = 0$). Thus a lower bound $\underline{\mu}$ for the barrier parameter (parameter `ETA5` in subroutine `PMAX`) is used.

The efficiency of the primal interior point method is also sensitive to the way in which the barrier parameter decreases. Subroutine `PMAX` uses the formula

$$\mu_{k+1} = \max\left(\tilde{\mu}_{k+1},\, \underline{\mu}\right),$$

where

$$\tilde{\mu}_{k+1} = \min\left[\max(\lambda\mu_k,\, \mu_k/(100\mu_k+1)),\, \max(\|g_{\mu_k}(x_k)\|^2,\, 10^{-2k})\right]$$

and where λ is the rate of the barrier parameter decrease (parameter `ETA4` in subroutine `PMAX`).

Subroutine `PMAX` serves for minimization of three particular functions. If `IEXT` < 0, then function (24) is considered. If `IEXT` $= 0$, then

$$F(x) = \max_{1\le i\le n_a} |f_i(x)| = \max_{1\le i\le n_a} \left[\max(f_i(x), -f_i(x))\right].$$

If `IEXT` > 0, then

$$F(x) = \max_{1\le i\le n_a}(-f_i(x)).$$

7. Primal interior-point methods for l_1 optimization

Consider a function of the form

$$F(x) = \sum_{i=1}^{n_a} |f_i(x)| \tag{28}$$

(a sum of absolute values), where $f_i(x)$, $1 \le i \le n_a$ (n_a is usually large), are smooth functions depending on a small number of variables (n_i, say). In this case, the Jacobian matrix $J(x) = [J_{ij}(x)] = [\partial f_i(x)/\partial x_j]$ is sparse. The sparsity pattern of the Jacobian matrix is stored using arrays `IAG` and `JAG` in the way described in Section 4.

Primal interior point methods for l_1 optimization, proposed in [18], are based on three basic ideas. First, minimization of F is equivalent to the nonlinear programming problem with $n + n_a$ variables $x \in R^n$, $z \in R^{n_a}$:

$$\text{minimize} \quad \sum_{i=1}^{n_a} z_i \quad \text{subject to} \quad -z_i \le f_i(x) \le z_i, \quad 1 \le i \le n_a. \tag{29}$$

Secondly, this constrained problem is replaced by a sequence of unconstrained problems

$$\begin{aligned} \text{minimize} \quad B_\mu(x,z) &= \sum_{i=1}^{n_a} z_i - \mu \sum_{i=1}^{n_a} \log(z_i - f_i(x)) - \mu \sum_{i=1}^{n_a} \log(z_i + f_i(x)) \\ &= \sum_{i=1}^{n_a} z_i - \mu \sum_{i=1}^{n_a} \log(z_i^2 - f_i^2(x)) \end{aligned}$$

where $z_i > |f_i(x)|$, $1 \le i \le n_a$, and $\mu > 0$ (we assume that $\mu \to 0$ monotonically). Finally, the extra variables z_i, $1 \le i \le n_a$, are eliminated by solving the set of quadratic equations

$$\frac{2\mu z_i}{z_i^2 - f_i^2(x)} = 1, \quad 1 \le i \le n_a,$$

which follow from the first-order necessary conditions for the barrier function $B_\mu(x,z)$ with fixed $x \in R^n$. Solutions of the above equations define a vector $z_\mu(x) \in R^{n_a}$, where

$$z_\mu(x)_i = e_i^T z_\mu(x) = \mu + \sqrt{\mu^2 + f_i^2(x)}, \quad 1 \le i \le n_a. \tag{30}$$

This approach leads to inexact unconstrained minimizations of functions

$$B_\mu(x) = B_\mu(x, z_\mu(x)) = \sum_{i=1}^{n_a} [(z_\mu(x)_i - \mu \log(z_\mu(x)_i)] - n_a \mu \log(2\mu) \tag{31}$$

for suitable values of μ using the fact that

$$\nabla B_\mu(x) = g_\mu(x) = J^T(x) u_\mu(x)$$

and

$$\nabla^2 B_\mu(x) = G_\mu(x) + J^T(x) V_\mu(x) J(x),$$

where

$$u_\mu(x) = \begin{bmatrix} f_1(x)/z_\mu(x)_1 \\ \dots \\ f_{n_a}(x)/z_\mu(x)_{n_a} \end{bmatrix}, \tag{32}$$

$$G_\mu(x) = \sum_{i=1}^{n_a} e_i^T u_\mu(x) \nabla^2 f_i(x),$$

$$V_\mu(x) = \operatorname{diag}\left(2\mu/(z_\mu(x)_1^2 + f_1^2(x)), \dots, 2\mu/(z_\mu(x)_{n_a}^2 + f_{n_a}^2(x))\right) \tag{33}$$

(note that differences $z_i^2 - f_i^2(x)$, $1 \le i \le m$, sensitive to round-off errors, are not used in formulas (30)–(33)). The Hessian matrix $\nabla^2 B_\mu(x)$ of the barrier function $B_\mu(x)$ is positive definite if $G_\mu(x)$ (the Hessian matrix of the Lagrangian function) is positive definite.

Subroutine `PSUM` is based on a trust-region realization of the Newton-like methods. The Dennis–Mei (dog-leg) method described in Section 3 is used for computation of the trust-region step d using the quadratic model

$$Q(d) = \frac{1}{2} d^T \nabla^2 B_\mu(x) d + g_\mu^T(x) d$$

(more details are given in [18]). In fact, we use an approximation of $\nabla^2 B_\mu(x)$, such that $G_\mu(x)$ is determined either by partitioned variable metric updates described in Section 4 (if $\texttt{MED} = \texttt{1}$) or by gradient differences as in [5] (if $\texttt{MED} = \texttt{2}$). If $\texttt{MED} = \texttt{1}$, then we define reduced approximations of the Hessian matrices $\tilde{G}_i = Z_i^T G_i Z_i$, $1 \le i \le n_a$, as in Section 4. New reduced approximations of the Hessian matrices are computed by the formulas (27) described in Section 6.

A very important part of the primal interior point method is an update of the barrier parameter μ. There are two requirements, which play opposite roles. First, $\mu \to 0$ should hold, since this is the main property of every interior-point method. On the other hand, matrix $\nabla^2 B_\mu(x)$ can be very ill-conditioned when μ is too small. Thus a lower bound $\underline{\mu}$ for the barrier parameter (parameter `ETA5` in subroutine `PSUM`) is used.

The efficiency of the primal interior point method is also sensitive to the way in which the barrier parameter decreases. Subroutine `PSUM` uses the formula

$$\mu_{k+1} = \max(\underline{\mu}, \|g_k(x_k)\|^2) \quad \text{if} \quad \rho(d_k) \ge \underline{\rho} \quad \text{and} \quad \|g_k(x_k)\|^2 \le \mu_k/100,$$

and $\mu_{k+1} = \mu_k$ otherwise ($\rho(d_k)$ and $\underline{\rho}$ are defined in Section 3).

8. Methods for sparse systems of nonlinear optimization

Consider the system of nonlinear equations

$$f(x) = 0, \tag{34}$$

where $f : R^n \to R^n$ is a continuously differentiable mapping and assume that the Jacobian matrix $J(x) = [J_{ij}(x)] = [\partial f_i(x)/\partial x_j]$ is sparse. The sparsity pattern of the Jacobian matrix is stored using arrays `IAG` and `JAG` in the way described in Section 4 (where $n_a = n$). Let A be an approximation of the Jacobian matrix $J = J(x)$ and let $F = F(x) = (1/2)\|f(x)\|^2$. Methods considered in this section are realized in the line-search framework. They generate a sequence of points $x_k \in R^n$, $k \in N$, such that

$$x_{k+1} = x_k + \alpha_k d_k, \quad k \in N, \tag{35}$$

where $d_k \in R^n$ is a direction vector determined as an approximate solution of the linear system $A_k d + f_k = 0$ such that

$$\|A_k d_k + f_k\| \le \overline{\omega}_k \|f_k\| \tag{36}$$

with the precision $0 \le \overline{\omega}_k \le \overline{\omega} < 1$ and α_k is the step-size chosen in such a way that it is the first member of the sequence α_k^j, $j \in N$, where $\alpha_k^1 = 1$ and $\underline{\beta}\alpha_k^j \le \alpha_k^{j+1} \le \overline{\beta}\alpha_k^j$ with $0 < \underline{\beta} \le \overline{\beta} < 1$, satisfying

$$F_{k+1} - F_k \le -\varepsilon_1 \alpha_k f_k^T A_k d_k, \tag{37}$$

with the line search parameter $0 < \varepsilon_1 < 1/2$. We use the values $\underline{\beta} = 0.1$, $\overline{\beta} = 0.9$ and $\varepsilon_1 = 10^{-4}$ in our subroutines. The value α_k^{j+1} can be determined by a bisection (`MES` $= 1$) or by a two-point quadratic interpolation (`MES` $= 2$) or by a three-point quadratic interpolation (`MES` $= 3$) or by a three-point cubic interpolation (`MES` $= 4$).

To obtain a superlinear rate of convergence, the condition $\overline{\omega}_k \to 0$ has to be satisfied. Therefore, we set

$$\overline{\omega}_k = \min(\max(\|f_k\|^\nu, \gamma(\|f_k\|/\|f_{k-1}\|)^\alpha), 1/k, \overline{\omega}),$$

where $\nu = 1/2$, $\gamma = 1$, $\alpha = (1+\sqrt{5})/2$ and $\overline{\omega} = 1/2$.

If $A_k \ne J_k$, then a safeguard based on restarts is used. It consists in setting $A_{k+1} = J_{k+1}$ if $j > \underline{j}$ or $A_k = J_k$ (with repeating the k-th iteration) if $j > \overline{j}$, where $0 < \underline{j} < \overline{j}$. We use the values $\underline{j} = 1$ and $\overline{j} = 5$. The restart of the form $A_k = J_k$ is also used whenever

$$-d_k^T J_k^T f_k \le \underline{\varepsilon} \|d_k\| \|J_k^T f_k\|,$$

where $0 < \underline{\varepsilon} < 1$ is a restart tolerance (we use the value $\underline{\varepsilon} = 10^{-12}$ in our subroutines).

The direction vector d_k (an approximate solution of the linear system $A_k d + f_k = 0$) is determined by using the preconditioned smoothed CGS method described in [34]. To simplify the description of this method, we omit the outer index k and denote the inner index by i. Let $h = A^T f$. We set $s_1 = 0$, $\overline{s}_1 = 0$, $r_1 = f$, $\overline{r}_1 = f$, $p_1 = f$, $u_1 = f$ and for $i = 1, 2, 3, \ldots$ we proceed in the following way. If $\|r_i\| \le \overline{\omega}\|f\|$, then set $d = s_i$ and terminate the process. Otherwise compute

$$v_i = AC^{-1}p_i, \quad \alpha_i = h^T \overline{r}_i / h^T v_i,$$

$$\begin{aligned}
q_i &= u_i - \alpha_i v_i, \\
\overline{s}_{i+1} &= \overline{s}_i + \alpha_i C^{-1}(u_i + q_i), \\
\overline{r}_{i+1} &= \overline{r}_i + \alpha_i A C^{-1}(u_i + q_i), \quad \beta_i = h^T \overline{r}_{i+1} / h^T \overline{r}_i, \\
u_{i+1} &= \overline{r}_{i+1} + \beta_i q_i, \quad p_{i+1} = u_{i+1} + \beta_i (q_i + \beta_i p_i), \\
[\lambda_i, \mu_i]^T &= \arg \min_{[\lambda,\mu]^T \in R^2} \|\overline{r}_{i+1} + \lambda(r_i - \overline{r}_{i+1}) + \mu v_i\|, \\
s_{i+1} &= \overline{s}_{i+1} + \lambda_i (s_i - \overline{s}_{i+1}) + \mu_i C^{-1} p_i, \\
r_{i+1} &= \overline{r}_{i+1} + \lambda_i (r_i - \overline{r}_{i+1}) + \mu_i v_i.
\end{aligned}$$

The matrix C serves as a preconditioner. The choice $C = I$ is used if $\texttt{MOS2} = 1$ or C is defined as an incomplete LU decomposition of the matrix A if $\texttt{MOS2} = 2$ ($\texttt{MOS2}$ is a parameter of the subroutines `PEQN` and `PEQL`). If $\texttt{MOS2} = 3$, a preliminary solution obtained by the incomplete LU decomposition can be accepted. In this case, we first compute vectors $d_1 = -C^{-1} f$, $r_1 = A d_1 + f$. If $\|r_1\| \leq \overline{\omega} \|f\|$, then we set $d = d_1$ and terminate the process, otherwise we continue by CGS iterations as above.

More details concerning globally convergent line-search methods for systems of nonlinear equations can be found in [22].

8.1 Inexact discrete Newton method

Subroutine `PEQN` is an implementation of the inexact discrete Newton method. This simple method is based on the elementwise differentiation. We always set $A_k = J(x_k)$, where

$$J_{ij}(x) = \frac{f_i(x + \delta_j e_j) - f_i(x)}{\delta_j} \tag{38}$$

for all pairs (i, j) corresponding to structurally nonzero elements of $J(x)$. Thus we need m scalar function evaluations (i.e. m/n equivalent vector function evaluations), where m is the number of structurally nonzero elements of $J(x)$.

Nonzero elements of sparse Jacobian matrix $J(x)$ can be also derived by using the groupwise differentiation [4]. Since our comparative tests have shown that the efficiencies of both these approaches are practically the same (see [22]), we use the simpler elementwise differentiation in our subroutines.

8.2 Inverse column-update quasi-Newton method

Subroutine `PEQL` is an implementation of the inverse column update method, which is introduced in [24]. This method uses an approximation $S_k = A_k^{-1}$ of the inverse Jacobian matrix J_k^{-1}. Therefore, we simply set $d_k = -S_k f_k$ if a restart is not used. Denote by

$$\begin{aligned}
&s_k = x_{k+1} - x_k, \quad s_{k-1} = x_k - x_{k-1}, \; \ldots, \; s_{k-m} = x_{k-m+1} - x_{k-m}, \\
&y_k = f_{k+1} - f_k, \quad y_{k-1} = f_k - f_{k-1}, \; \ldots, \; y_{k-m} = f_{k-m+1} - f_{k-m}
\end{aligned}$$

the last m differences of points and function vectors, respectively, where the lower index $k - m$ corresponds to the iteration with the restart. Let

$$|e_{i_{k-1}}^T y_{k-1}| = \max_{1 \leq i \leq n} |e_i^T y_{k-1}|, \ldots, |e_{i_{k-m}}^T y_{k-m}| = \max_{1 \leq i \leq n} |e_i^T y_{k-m}|$$

(where e_i, $1 \leq i \leq n$, are columns of the unit matrix). Then the vector $S_k f_k$ can be computed by the formula

$$S_k f_k = S_{k-m} f_k + \frac{e_{i_{k-1}}^T f_k}{e_{i_{k-1}}^T y_{k-1}} v_{k-1} + \ldots + \frac{e_{i_{k-m}}^T f_k}{e_{i_{k-m}}^T y_{k-m}} v_{k-m},$$

where $v_{k-1} = d_{k-1} - S_{k-1} y_{k-1}$, …, $v_{k-m} = d_{k-m} - S_{k-m} y_{k-m}$ are vectors computed recursively by the formula

$$v_k = d_k - S_k y_k = d_k - S_{k-m} y_k - \frac{e_{i_{k-1}}^T y_k}{e_{i_{k-1}}^T y_{k-1}} v_{k-1} - \ldots - \frac{e_{i_{k-m}}^T y_k}{e_{i_{k-m}}^T y_{k-m}} v_{k-m}.$$

In both of these formulae we use the matrix $S_{k-m} = (L_{k-m} U_{k-m})^{-1}$, where $L_{k-m} U_{k-m}$ is the incomplete LU decomposition of the Jacobian matrix $J(x_{k-m})$ computed by (38). Note that the vectors $e_{i_{k-1}}, \ldots, e_{i_{k-m}}$ do not need to be stored. We only use indices of their unique nonzero elements. The limited memory column update method needs to be restarted periodically after $\overline{m}$ iterations (parameter `MF` in the subroutine `PEQL`), since at most $\overline{m}$ vectors can be stored.

REFERENCES

[1] Al-Baali M., Fletcher R.: Variational methods for nonlinear least squares. Journal of Optimization Theory and Applications 36 (1985) 405-421.

[2] Brown, P.N., Saad, Y.: Convergencet theory of nonlinear Newton-Krylov algorithms. SIAM Journal on Optimization 4 (1994) 297-330.

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

[4] Coleman T.F., and Moré J.J.: Estimation of sparse Jacobian and graph coloring problem. SIAM Journal on Numerical Analysis 20, (1983) 187-209.

[5] Coleman, T.F., Moré J.J.: Estimation of sparse Hessian matrices and graph coloring problems. Mathematical Programming 28 (1984) 243-270.

[6] Curtis, A.R., Powell, M.J.D., and Reid, J.K.: On the estimation of sparse Jacobian matrices. IMA Journal of Aplied Mathematics 13 (1974) 117-119.

[7] Dembo, R.S, Eisenstat, S.C., and Steihaug T.: Inexact Newton methods. SIAM J. on Numerical Analysis 19 (1982) 400-408.

[8] Dembo, R.S, Steihaug T.: Truncated Newton algorithms for large-scale optimization. Math. Programming 26 (1983) 190-212.

[9] Dennis J.E., Mei H.H.W: An unconstrained optimization algorithm which uses function and gradient values. Report No. TR 75-246, 1975.

[10] Gill P.E., Murray W.: Newton type methods for unconstrained and linearly constrained optimization. Math. Programming 7 (1974) 311-350.

[11] Gould N.I.M, Lucidi S., Roma M., Toint P.L.: Solving the trust-region subproblem using the Lanczos method. Report No. RAL-TR-97-028, 1997.

[12] Griewank A., Toint P.L.: Partitioned variable metric updates for large-scale structured optimization problems. Numer. Math. 39 (1982) 119-137.

[13] Liu D.C., Nocedal J.: On the limited memory BFGS method for large scale optimization. Math. Programming 45 (1989) 503-528.

[14] Lukšan L.: Combined trust region methods for nonlinear least squares. Kybernetika 32 (1996) 121-138.

[15] Lukšan L.: Hybrid methods for large sparse nonlinear least squares. J. Optimizaton Theory and Applications 89 (1996) 575-595.

[16] Lukšan L., Matonoha C., Vlček J.: A shifted Steihaug-Toint method for computing a trust-region step. Report V-914, Prague, ICS AS CR, 2004.

[17] Lukšan L., Matonoha C., Vlček J.: Primal interior-point method for large sparse minimax optimization. Technical Report V-941, Prague, ICS AS CR, 2005.

[18] Lukšan L., Matonoha C., Vlček J.: Trust-region interior point method for large sparse l_1 optimization. Technical Report V-942, Prague, ICS AS CR, 2005.

[19] Lukšan L., Spedicato E.: Variable metric methods for unconstrained optimization and nonlinear least squares. Journal of Computational and Applied Mathematics 124 (2000) 61-93.

[20] Lukšan L., Tůma M., Hartman J., Vlček J., Ramešová N., Šiška M., Matonoha C.: Interactive System for Universal Functional Optimization (UFO). Version 2006. Technical Report V-977. Prague, ICS AS CR 2006.

[21] Lukšan L., Vlček J. Sparse and partially separable test problems for unconstrained and equality constrained optimization. Report V-767, Prague, ICS AS CR, 1998.

[22] Lukšan L., Vlček J.: Computational Experience with Globally Convergent Descent Methods for Large Sparse Systems of Nonlinear Equations. Optimization Methods and Software 8 (1998) 201-223.

[23] Lukšan L., Vlček J.: Variable metric method for minimization of partially separable nonsmooth functions. Pacific Journal on Optimization 2 (2006).

[24] Martinez, J.M., and Zambaldi, M.C.: An Inverse Column-Updating Method for Solving Large-Scale Nonlinear Systems of Equations. Optimization Methods and Software 1 (1992) 129-140.

[25] Moré J.J., Sorensen D.C.: Computing a trust region step. SIAM Journal on Scientific and Statistical Computations 4 (1983) 553-572.

[26] Nocedal J.: Updating quasi-Newton matrices with limited storage. Math. Comp. 35 (1980) 773-782.

[27] Nowak U., Weimann L.: A family of Newton codes for systems of highly nonlinear equations. Tech. Report TR-91-10, Konrad-Zuse-Zentrum für Informationstechnik, Berlin 1991.

[28] Pernice M., Walker H.F.: NITSOL: A Newton iterative solver for nonlinear systems. SIAM J. on Scientific Computing 19 (1998) 302-318.

[29] Schlick T., Fogelson A.: TNPACK – A Truncated Newton Minimization Package for Large-Scale Problems (Parts I and II). ACM Transactions on Mathematical Software 18 (1992) 46-111.

[30] Steihaug T.: The conjugate gradient method and trust regions in large-scale optimization. SIAM Journal on Numerical Analysis 20 (1983) 626-637.

[31] Toint P.L.: Towards an efficient sparsity exploiting Newton method for minimization. In: Sparse Matrices and Their Uses (I.S.Duff, ed.), Academic Press, London 1981, 57-88.

[32] Toint P.L.: Subroutine VE08. Harwell Subroutine Library. Specifications (Release 12), Vol, 2, AEA Technology, December 1995, 1162-1174.

[33] Toint P.L.: Subroutine VE10. Harwell Subroutine Library. Specifications (Release 12), Vol, 2, AEA Technology, December 1995, 1187-1197.

[34] Tong C.H.: A comparative study of preconditioned Lanczos methods for nonsymmetric linear systems. Sandia National Laboratories, Sandia Report No. SAND91-8240B, Sandia National Laboratories, Livermore, 1992.

[35] Tůma M.: A note on direct methods for approximations of sparse Hessian matrices. Aplikace Matematiky 33 (1988) 171-176.

[36] Vlček J., Lukšan L.: Globally convergent variable metric method for nonconvex nondifferentiable unconstrained minimization. Journal of Optimization Theory and Applications 111 (2001) 407-430.

[37] Watson L.T., Melville R.C., Morgan A.P., Walker H.F: Algorithm 777. HOMPACK90: A Suite of Fortran 90 Codes for Globally Convergent Homotopy Algorithms. ACM Transactions on Mathematical Software 23 (1997) 514-549.