Optimization Methods and Software

ISSN: 1055-6788 (Print) 1029-4937 (Online) Journal homepage: https://www.tandfonline.com/loi/goms20

A trust-region derivative-free algorithm for constrained optimization

P.D. Conejo, E.W. Karas & L.G. Pedroso

To cite this article: P.D. Conejo, E.W. Karas & L.G. Pedroso (2015) A trust-region derivative-free algorithm for constrained optimization, Optimization Methods and Software, 30:6, 1126-1145, DOI: 10.1080/10556788.2015.1026968

To link to this article: https://doi.org/10.1080/10556788.2015.1026968

Published online: 05 May 2015.

Optimization Methods & Software, 2015
Vol. 30, No. 6, 1126–1145, http://dx.doi.org/10.1080/10556788.2015.1026968

A trust-region derivative-free algorithm for constrained optimization

P.D. Conejo[a*], E.W. Karas[b] and L.G. Pedroso[b]

[a]*State University of West Paraná, Center of Exact Sciences, Cascavel, PR, Brazil;* [b]*Department of Mathematics, Federal University of Paraná, Curitiba, PR, Brazil*

(*Received 14 February 2014; accepted 1 March 2015*)

We propose a trust-region algorithm for constrained optimization problems in which the derivatives of the objective function are not available. In each iteration, the objective function is approximated by a model obtained by quadratic interpolation, which is then minimized within the intersection of the feasible set with the trust region. Since the constraints are handled in the trust-region subproblems, all the iterates are feasible even if some interpolation points are not. The rules for constructing and updating the quadratic model and the interpolation set use ideas from the BOBYQA software, a largely used algorithm for box-constrained problems. The subproblems are solved by ALGENCAN, a competitive implementation of an Augmented Lagrangian approach for general-constrained problems. Some numerical results for the Hock–Schittkowski collection are presented, followed by a performance comparison between our proposal and three derivative-free algorithms found in the literature.

Keywords: derivative-free optimization; trust-region algorithms; constrained optimization problems; numerical experiments

AMS Subject Classification: 90C56; 65K05; 49M37

1. Introduction

We propose a trust-region algorithm for solving the problem

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in \Omega \end{array} \tag{1}$$

with $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ and $\Omega = \{x \in \mathbb{R}^n \mid h(x) = 0, g(x) \leq 0\}$ where $h : \mathbb{R}^n \to \mathbb{R}^p$ and $g : \mathbb{R}^n \to \mathbb{R}^q$ are differentiable functions. Although our algorithm can be applied when the objective function is nonsmooth, it was designed for the class of problems in which f is smooth but its derivatives are not available.

Derivative-free methods for continuous optimization have received increasing attention from the scientific community due to the wide range of problems where derivatives cannot be explicitly computed [9,13], as for example when the objective function is provided by a simulation package or a black box. Considering the class of trust-region derivative-free methods, many algorithms can be found in the literature. For the unconstrained case, i.e. when $\Omega = \mathbb{R}^n$, Conn

*Corresponding author. Email: pconejo33@gmail.com *et al.* [8,12,13], Scheinberg and Toint [29] and Powell [27] presented algorithms with global convergence results, while Fasano *et al.* [16], Conn and Toint [14], and Powell [24,25] discussed algorithms with good practical performance. It is worth pointing out that Powell has contributed largely in the area of derivative-free methods. For problems with box constraints, he proposed the box constraints, he proposed the `BOBYQA` software [21,26] and has recently released the `LINCOA` software [22,28] for linearly constrained problems. Arouxét *et al.* [3] and Gratton *et al.* [18] proposed active-set algorithms for bound-constrained optimization. For general constrained problems, which is the focus of the present paper, it is important to mention the algorithm introduced by Conn *et al.* [9]. The implementation of this algorithm, called `DFO`, is competitive even when compared to very recent approaches. Regarding theoretical results, Conejo *et al.* [6] proposed a globally convergent trust-region algorithm for convex constrained problems.

The classical trust-region framework [7] for unconstrained problems consists in approximating at each iteration the objective function by a quadratic model and minimize it within the trust-region, which is a ball centred at the current iterate with suitable radius. In derivative-free techniques, the models are usually constructed by quadratic interpolation [5,10,13] or by polynomial regression [11]. Our approach, based on quadratic interpolation, uses the structure presented in the `BOBYQA` software [26] for constructing and updating the model and the interpolation set. As in the BOBYQA algorithm, there is freedom in the quantity of interpolation points, which is chosen by the user and fixed through the iterations. This freedom would result in the non-unicity of the interpolation model. However, among all the models that interpolate the objective function in the current interpolation set, we choose the one with the Hessian which is closest to that of the previous model in terms of the Frobenius norm, and with this choice the model is uniquely defined. The subproblem, that consists in minimizing the quadratic model within the intersection of the feasible set and the trust region, is solved by ALGENCAN, which is an implementation of the Augmented Lagrangian algorithm proposed in [1,2]. Thus the constraints are treated in the subproblems and the algorithm generates a sequence of feasible iterates. A new iterate is incorporated in the interpolation set whenever it provides a decrease in the objective function.

The paper is organized as follows. In Section 2, we present our trust-region derivative-free algorithm. Section 3 discusses the construction and update of the interpolation set and model. Numerical results are presented in Section 4. Finally, we state some conclusions.

Notation: The symbol $\mathcal{P}_2^n$ denotes the set of all quadratic polynomials in $\mathbb{R}^n$; e^i denotes the ith coordinate vector; $\|\cdot\|$ denotes the Euclidean norm and $Q(Y) = f(Y)$ means that $Q(y^i) = f(y^i)$ for all $y^i \in Y$.

2. The algorithm

In this section, we state our algorithm, postponing to the next section the discussion about the details of its steps.

At each iteration k, we construct a quadratic model Q_k by polynomial interpolation. Given $x^b \in \mathbb{R}^n$, the quadratic model assumes the form

$$Q_k(x) = c_k + g_k^T(x - x^b) + \tfrac{1}{2}(x - x^b)^T G_k(x - x^b), \tag{2}$$

where $c_k \in \mathbb{R}$, $g_k \in \mathbb{R}^n$ and $G_k \in \mathbb{R}^{n\times n}$ is a symmetric matrix.

The number of interpolation points,

$$m \in [n+2, \bar{m}] \quad \text{with } \bar{m} = \tfrac{1}{2}(n+1)(n+2), \tag{3}$$

is chosen by the user and fixed through the iterations. We denote the current interpolation set by $Y_k = \{y^1, y^2, \dots, y^m\} \subset \mathbb{R}^n$.

As usual in trust-region frameworks, our algorithm considers a sequence of subproblems consisting in minimizing the model Q_k within the trust region with radius $\Delta_k > 0$ centred at the current point x_k. Furthermore, the constraints of problem (1) are incorporated in the subproblem, that takes the form

$$\begin{aligned} \text{minimize} \quad & Q_k(x) \\ \text{subject to} \quad & x \in \Omega, \\ & \|x - x_k\|_\infty \leq \Delta_k. \end{aligned} \tag{4}$$

Although the Euclidean norm is typically used to define the trust region, we adopted the ℓ_∞-norm to convert the trust region into box constraints.

The algorithm proposed in this paper is defined as follows.

ALGORITHM 1

```
*Data:* $x^1 \in \Omega$, $\varepsilon > 0$, $\rho_1 > \varepsilon$, $\gamma \in (0,1)$, $m \in [n+2, \bar{m}]$, $s > 2$, $\Delta_1 = \rho_1$.
$k = 1$.
**Step 1:** Constructing the model
    *Set* $x^b = x_k$.
    *Construct the interpolation set* $Y_k$ *with* $y^1 = x_k$ *and* $\rho = \rho_k$.
    *Construct the model* $Q_k$ *by polynomial interpolation.*
**Step 2:** Solving the subproblem
    *Compute* $x^+$, *solution of the subproblem* (4).
**Step 3:** Updates
    IF $\|x^+ - x_k\| \leq 0.5\rho_k$, THEN
        IF $\rho_k \leq \varepsilon$, THEN STOP.
        IF $\max_{1\leq j\leq m}\{\|y^j - x_k\|\} > s\rho_k$, THEN $\rho_{k+1} = \rho_k$.
        ELSE, $\rho_{k+1} = \gamma\rho_k$.
        $x_{k+1} = x_k$, $\Delta_{k+1} = \Delta_k$, $k = k+1$ *and go to* Step 1.
    ELSE,
        $r = \frac{f(x_k) - f(x^+)}{Q_k(x_k) - Q_k(x^+)}$,
        $\bar{\Delta} = \begin{cases} 0.5\Delta_k, & \text{if} \quad r < 0.1, \\ \Delta_k, & \text{if} \quad 0.1 \leq r \leq 0.7, \\ 2\Delta_k, & \text{if} \quad r > 0.7. \end{cases}$
        IF $f(x^+) < f(x_k)$, THEN
            *Choose* $y^t$ *that will leave the interpolation set* $Y_k$.
            *Set* $Y^+ = Y_k \backslash \{y^t\} \cup \{x^+\}$.
        IF $r \geq 0.1$, THEN
            $x_{k+1} = x^+$, $\Delta_{k+1} = \bar{\Delta}$, $\rho_{k+1} = \rho_k$, $Y_{k+1} = Y^+$.
            *Compute* $Q_{k+1}$ *by solving* (6), $k = k+1$ *and go to* Step 2.
        ELSE,
            IF $Y^+$ *is not sufficiently poised or* $\max_{1\leq j\leq m}\{\|y^j - x_k\|\} > s\rho_k$, THEN
                $x_{k+1} = x_k$, $\Delta_{k+1} = \bar{\Delta}$, $\rho_{k+1} = \rho_k$, $k = k+1$ *and go to* Step 1.
            ELSE,
                IF $\rho_k \leq \varepsilon$, THEN STOP.
                ELSE,
        IF $f(x^+) < f(x_k)$, THEN
            $Y_{k+1} = Y^+$ and compute $Q_{k+1}$ by solving (6).
        ELSE,
            $Y_{k+1} = Y_k$, $Q_{k+1} = Q_k$.
    $x_{k+1} = x_k$, $\Delta_{k+1} = \rho_k$, $\rho_{k+1} = \gamma\rho_k$, $k = k+1$ and go to Step 2.
```

Note that the sequence (x_k) generated by the algorithm is feasible, but the interpolation points may be infeasible. The point x^+, obtained as a solution of the subproblem (4), will be incorporated to the interpolation set whenever it provides a simple decrease in the objective function, unless the resulting interpolation set is not sufficiently poised. The meaning of the expression *sufficiently poised* will be explained in the next section. If the decrease at x^+ is sufficient, in the sense that $r \geq 0.1$, then this point is accepted as the new iterate. The scalar r is the ratio between the actual reduction and the one predicted by the model, which is a classical measure in trust-region algorithms. The magnitude of this scalar is also used to decide if the trust-region radius Δ_k will be reduced, maintained or increased. The parameter $\rho_k \leq \Delta_k$ controls the diameter of the interpolation set and is related to the stopping criterion. If $\|x^+ - x_k\| < 0.5\rho_k$ or $r < 0.1$, the progress in the current iteration is considered insufficient. In this situation, the algorithm might test if it is time to stop, since the iterate may be close to a solution. If the algorithm does not stop, at least one of the following actions will take place: the parameter ρ_k will be reduced, so the next interpolation sets will tend to have smaller diameters, resulting on more accurate models, or the interpolation set will be redefined and the model reconstructed at Step 1, since the lack of progress may be due to accumulation of errors from the previous model updates. In order to avoid unnecessary reductions of ρ_k, we perform the test

$$\max_{1\leq j\leq m}\{\|y^j - x_k\|\} > s\rho_k.$$

If this condition holds, at least one of the interpolation points is far from the current iterate, then the value of ρ_k is kept for the next iteration and the interpolation set is reconstructed at Step 1.

3. The model and the interpolation set

In this section, we discuss how to construct and update the interpolation set and the model. We would like to emphasize that the procedures discussed in this section are based on [26]. Consider the current interpolation set $Y_k = \{y^1, y^2, \ldots, y^m\}$. If $m = \bar{m}$ and Y_k is poised [13], the interpolation conditions

$$Q_k(Y_k) = f(Y_k) \tag{5}$$

determine uniquely the model Q_k defined in (2). On the other hand, when $m < \bar{m}$, there might be infinitely many models that verify (5).

In some iterations, the model and the interpolation set are constructed from scratch, while in others they are updated from the previous ones. The construction takes place in the first iteration and whenever a lack of progress is detected in the algorithm. In these cases, the interpolation points are chosen differing from the current point in at most two components, so a quadratic model can be easily computed.

During an updating procedure, only one interpolation point of Y_k, say y^t, is modified and replaced by x^+, so $Y_{k+1} = Y_k \backslash \{y^t\} \cup \{x^+\}$. When computing Q_{k+1} from Q_k, among all the models that satisfy the interpolation conditions, we choose the one that is as close as possible in the

Frobenius norm to the current model Q_k, which means that Q_{k+1} is the solution of

$$\begin{aligned} \underset{Q\in\mathcal{P}_2^n}{\text{minimize}} \quad & \|\nabla^2 Q - G_k\|_F \\ \text{subject to} \quad & Q(Y_{k+1}) = f(Y_{k+1}). \end{aligned} \tag{6}$$

3.1 *Construction*

The interpolation set $Y_k = \{y^1, y^2, \dots, y^m\}$ and the model Q_k are constructed according to rules similar to the ones found in [26]. In our algorithm, these constructions take place at Step 1, which is called not only at the first iteration, but whenever it is detected that the interpolation set has to be corrected. For reconstructing Y_k, Powell uses a procedure called `RESCUE` [26, Sec. 5], which tries to improve the geometry of the interpolation set replacing as few interpolation points as possible. Our algorithm replaces all interpolation points at a reconstruction, which is what the `RESCUE` procedure does in the worst case.

Given $y^1 \in \Omega$ and $\rho > 0$, the next $2n$ interpolation points are given by

$$y^{i+1} = y^1 + \rho e^i \quad \text{for } i = 1, \dots, n, \tag{7}$$

$$y^{n+i+1} = y^1 - \rho e^i \quad \text{for } i = 1, \dots, \min\{n, m-n-1\}. \tag{8}$$

The remaining $m - 2n - 1$ points are given by

$$y^j = y^1 + \rho e^{u_j} + \rho e^{v_j} \tag{9}$$

for $2n + 2 \le j \le m$, where

$$u_j = \begin{cases} j - 2n - 1, & 2n + 2 \le j \le 3n + 1, \\ u_{j-n}, & 3n + 2 \le j \le m \end{cases} \tag{10}$$

and

$$v_j = \begin{cases} u_j + c_j & \text{if } (u_j + c_j) \in \{1, \dots, n\} \\ u_j + c_j - n & \text{if } (u_j + c_j) \notin \{1, \dots, n\} \end{cases} \tag{11}$$

with $c_j = \lceil (j - 2n - 1)/n \rceil$, where $\lceil a \rceil$ denotes the smallest integer greater than or equal to a. It is worth noticing that this interpolation set is quite simple. It consists of a point y^1, at most $2n$ points in the coordinate directions from y^1 and $m - 2n - 1$ points differing from y^1 in two components.

Given the interpolation set $Y_k = \{y^1, y^2, \dots, y^m\}$ constructed according to (7)–(11), consider $x^b = y^1$. As discussed in [26], the quadratic model that satisfies the interpolation condition

$Q_k(Y_k) = f(Y_k)$ is given by (2) with

$$
\begin{aligned}
c_k &= f(y^1),\\
[g_k]_j &= \frac{1}{2\rho}(f(y^{j+1}) - f(y^{j+1+n})), \quad j = 1, \dots, \min\{n, m-n-1\},\\
[g_k]_j &= \frac{1}{\rho}(f(y^{j+1}) - f(y^1)), \quad j = m-n, \dots, n,\\
[G_k]_{j,j} &= \frac{1}{\rho^2}(f(y^{j+1}) + f(y^{j+1+n}) - 2f(y^1)), \quad j = 1, \dots, \min\{n, m-n-1\},\\
[G_k]_{u_j,v_j} &= \frac{1}{\rho^2}(f(y^j) - f(y^{u_j+1}) - f(y^{v_j+1}) + f(y^1)), \quad j = 2n+2, \dots, m,\\
[G_k]_{u_j,v_j} &= 0, \quad j = m+1, \dots, \bar{m},\\
[G_k]_{v_j,u_j} &= [G_k]_{u_j,v_j}, \quad j = 2n+2, \dots, \bar{m}.
\end{aligned}
$$

The model is constructed with little computational effort due to the simplicity of the constructed interpolation set. Note that the matrix G_k is diagonal if $m \in [n+2, 2n+1]$.

3.2 *Auxiliary parameters*

In this section, we define some parameters that are necessary for the update procedures. The first one is an auxiliary matrix $H \in \mathbb{R}^{(m+n+1)\times(m+n+1)}$. During a construction procedure, we define the matrix H as proposed in [24] by

$$
H = \left(\begin{array}{c|c} ZZ^T & E^T \\ \hline E & \Upsilon \end{array}\right) \tag{12}
$$

with $Z \in \mathbb{R}^{m\times(m-n-1)}$, $\Upsilon \in \mathbb{R}^{(n+1)\times(n+1)}$ and $E \in \mathbb{R}^{(n+1)\times m}$ given by

$$
\begin{aligned}
Z_{1,j} &= -\frac{\sqrt{2}}{\rho^2}, \quad Z_{j+1,j} = Z_{n+j+1,j} = \frac{\sqrt{2}}{2\rho^2}, \quad j = 1, \dots, \min\{n, m-n-1\},\\
Z_{1,j-n-1} &= Z_{j,j-n-1} = \frac{1}{\rho^2}, \quad j = 2n+2, \dots, m,\\
Z_{u_j+1,j-n-1} &= Z_{v_j+1,j-n-1} = -\frac{1}{\rho^2}, \quad j = 2n+2, \dots, m,\\
E_{1,1} &= 1, \quad E_{j,j} = \frac{1}{2\rho}, \quad E_{j,j+n} = -\frac{1}{2\rho}, \quad j = 2, \dots, \min\{n+1, m-n\},\\
E_{j,1} &= -\frac{1}{\rho}, \quad E_{j,j} = \frac{1}{\rho}, \quad j = m-n+1, \dots, n+1,\\
\gamma_{j,j} &= -\frac{\rho^2}{2}, \quad j = m-n+1, \dots, n+1,
\end{aligned}
$$

the remaining elements of Z, E and Υ are null.

Note that this matrix was not used in the previous section to construct neither the model nor the interpolation set, but it has to be defined since it will be necessary ahead. At an update, once defined t such that y^t will leave the interpolation set, the new matrix H^+ is obtained from the previous one as proposed in [24] by

$$H^+ = H + \frac{1}{\sigma}(\alpha(e^t - Hw)(e^t - Hw)^T - \beta He^t(e^t)^T H + \tau(He^t(e^t - Hw)^T + (e^t - Hw)(e^t)^T H)), \tag{13}$$

where the vector $w \in \mathbb{R}^{m+n+1}$ is also an auxiliary parameter, given by

$$\begin{aligned} w_j &= \tfrac{1}{2}((y^j - x^b)^T(x^+ - x^b))^2, \quad j = 1, 2, \ldots, m, \\ w_{m+1} &= 1, \\ w_{j+m+1} &= [x^+ - x^b]_j, \quad j = 1, 2, \ldots, n, \end{aligned} \tag{14}$$

with the parameters α, β, τ and σ assuming the values

$$\begin{aligned} \alpha &= (e^t)^T He^t, \\ \beta &= \tfrac{1}{2}\|x^+ - x^b\|^4 - w^T Hw, \\ \tau &= (e^t)^T Hw, \\ \sigma &= \alpha\beta + \tau^2. \end{aligned} \tag{15}$$

We point out that these auxiliary parameters, proposed by Powell, are crucial to our algorithm. Finding a model Q_k satisfying conditions (5) is not a very complicated task itself, yet it can demand much computational effort. Matrix H plays a very important role in this context, allowing us to update the model efficiently and with a low computational cost, as we will see in the next section.

3.3 *Update*

The update procedures for the interpolation set and the model are based on [26]. To update the interpolation set, first we choose a point y^t that will leave the set Y_k to be replaced by x^+. Assume that the current point x_k is at position $\bar{t}$ in the interpolation set Y_k, i.e. $x_k = y^{\bar{t}}$. The point y^t that will leave Y_k is such that

$$t = arg\,max_{j \in \{1,2,\ldots,m\} \setminus \{\bar{t}\}}\{\sigma(j)\}, \tag{16}$$

where

$$\sigma(j) = [H]_{j,j}\left(\frac{1}{2}\|x^+ - x^b\|^4 - w^T Hw\right) + ((e^j)^T Hw)^2$$

and the matrix H and the vector w are defined in the previous section.

If $\sigma(t) \leq \varepsilon_1$ for a fixed tolerance $\varepsilon_1 > 0$, the set $Y_k \setminus \{y^t\} \cup \{x^+\}$ is considered *not sufficiently poised*, in the sense that we would have a very small denominator in (13). In this case, the interpolation set and the model are reconstructed as discussed in Section 3.1.

Otherwise, the interpolation set for the next iteration will be $Y_{k+1} = Y_k \setminus \{y^t\} \cup \{x^+\}$. The new model Q_{k+1} is defined by the parameters c_{k+1}, g_{k+1} and G_{k+1} given by

$$c_{k+1} = c_k + c, \quad g_{k+1} = g_k + g \quad \text{and} \quad G_{k+1} = G_k + \sum_{j=1}^{m} \varphi_j(y_+^j - x^b)(y_+^j - x^b)^T,$$

with

$$\begin{pmatrix} \varphi \\ c \\ g \end{pmatrix} = (f(x^+) - Q_k(x^+))[H^+]_t. \tag{17}$$

In [23], it is shown that such model Q_{k+1} is the solution of (6).

4. Numerical results

Algorithm 1 was implemented in Fortran 77. The tests were performed using the gfortran compiler (32-bits), version gcc-4.2.3, in a notebook I7, 2.1GHz, 8GB of RAM. The algorithm was run with input $\rho_1 = 10^{-1}$, $\varepsilon = 10^{-4}$, $\varepsilon_1 = 10^{-10}$, $\gamma = 0.1$ and $s = 10$.

The subproblems in Step 2 were solved by ALGENCAN [1,2], a code for solving large-scale nonlinear programming problems periodically updated and available at the TANGO Project website http://www.ime.usp.br/~egbirgin/tango. We used ALGENCAN version 2.2.1 with all parameters set to their default values, except for epsfeas = epsopt = 10^{-8}. This choice implies that the sequence generated by the algorithm is feasible with tolerance 10^{-8}, in the sense that, for all k, $\psi(x_k) \le 10^{-8}$, where $\psi : \mathbb{R}^n \to \mathbb{R}$ is the infeasibility measure given by

$$\psi(x) = \max\{\|h(x)\|_\infty, \|g^+(x)\|_\infty\} \tag{18}$$

with $g_i^+(x) = \max\{0, g_i(x)\}$, for all $i = 1, \dots, q$.

If the provided initial point is infeasible with respect to Ω, we use ALGENCAN, starting on this point, to minimize the infeasibility measure $\psi^2/2$ in order to find $x_1 \in \Omega$.

We considered two variants of Algorithm 1. The first one uses $m = 5$ if $n = 2$ and $m = 2n + 3$ otherwise. This special choice for $n = 2$ is due to the fact that, in this case, $2n + 3$ is greater than the upper bound $\bar{m}$ defined in (3). The second variant uses $m = \bar{m}$ interpolation points.

To put our approach in perspective, we compared its performance with the following derivative-free algorithms found in the literature:

Inexact Restoration algorithm [4], a C++ code for equality-constrained problems, in which each iteration is decomposed into two phases: one that uses ALGENCAN for reducing a measure of infeasibility by means of an interface between C++ and Fortran, and another one that uses GSS to minimize a suitable objective function in a linear approximation of the feasible set.

DFO (Derivative-Free Optimization), a Fortran code based on the trust-region approach proposed in [9]. The models are constructed by quadratic interpolation using only feasible points. The subproblems are solved by the NPSOL algorithm [17].

LINCOA (Linearly Constrained Optimization Algorithm), a Fortran code [22] based on the trust-region approach proposed in [28]. LINCOA is an extension of BOBYQA [21,26] for linearly constrained optimization problems. The models are constructed by quadratic interpolation using $m \in [n + 2, \bar{m}]$ interpolation points. The linear constraints are treated by active sets.

Summing up, there are six solvers under analysis, which are:

- S_1: Algorithm 1 with $m = 5$ if $n = 2$ and $m = 2n + 3$ otherwise;
- S_2: Algorithm 1 with $m = \bar{m}$;
- IR: Derivative-free Inexact Restoration algorithm;
- DFO: DFO algorithm;
- L_1: LINCOA algorithm with $m = 5$ if $n = 2$ and $m = 2n + 3$ otherwise;
- L_2: LINCOA algorithm with $m = \bar{m}$.

For the numerical experiments, we considered the Hock–Schittkowski collection [19,31], which is a largely used test set for optimization algorithms. Although the derivatives of all functions that define the problems are available, the collection is often used to compare the

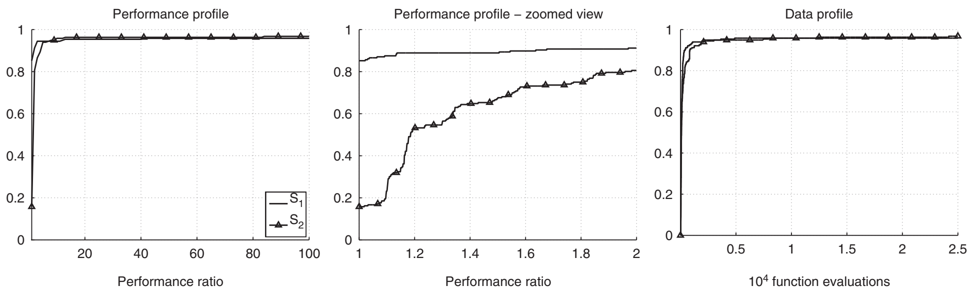

Figure 1. Performance and data profiles for the comparison between solvers S_1 and S_2.

performance of derivative-free algorithms. In our analysis, a point $\bar{x} \in \mathbb{R}^n$ is considered feasible if

$$\psi(\bar{x}) \leq 10^{-8}, \tag{19}$$

with ψ defined in (18). For comparing the algorithms, we used the data and performance profiles discussed in [15,20], considering the number of objective function evaluations as the performance measure, as usual in derivative-free optimization.

4.1 *Performance of solvers S_1 and S_2*

We tested the algorithm with all 216 problems of the Hock–Schittkowski collection [31] that involve at least one constraint beside box constraints. The dimension of the problems varies between 2 and 50 and the number of constraints between 1 and 45.

Similarly to [4], a problem is considered solved by an algorithm if it found a feasible point $\bar{x}$ satisfying

$$\frac{f(\bar{x}) - f_{HS}}{\max\{1, |f(\bar{x})|, |f_{HS}|\}} \leq 10^{-4}, \tag{20}$$

where f_{HS} is the solution available in the Hock–Schittkowski collection, found by the `NLPQLP` code [30].

Figure 1 shows the data and performance profiles related to the number of objective function evaluations. Both solvers S_1 and S_2 were competitive in terms of robustness, solving, respectively, 95.4% and 96.8% of the problems. On the other hand, S_1 was more efficient, since solvers S_1 and S_2 performed fewer function evaluations on 85.2% and 15.7% of the problems, respectively. Furthermore, when we allowed the algorithms to perform 500 objective function evaluations, S_1 solved 91.2% while S_2 solved 83.3% of the problems. The results of this section indicate that using less than $\bar{m}$ interpolation points provides better results in terms of efficiency.

4.2 *Comparison with solver* `IR`

In this section, we compare the performance of solvers S_1, S_2 and `IR`. We considered a subset of 210 problems from the 216 test problems of the last section. Problems 85, 284, 285, 383, 384, 392 were ignored because the `IR` algorithm exceeded 30 min of CPU time. The `IR` algorithm, whose code was kindly provided by the authors of [4], was run with default parameter values.

Similarly to [4], we say that an algorithm solved a problem if it found a feasible point $\bar{x}$ such that

$$\frac{f(\bar{x}) - f_{\min}}{\max\{1, |f(\bar{x})|, |f_{\min}|\}} \leq 10^{-4}, \tag{21}$$

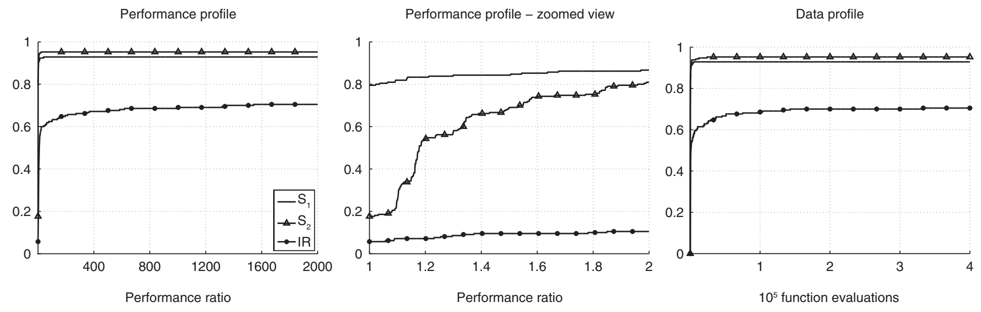

Figure 2. Performance and data profiles for solvers S_1, S_2 and IR.

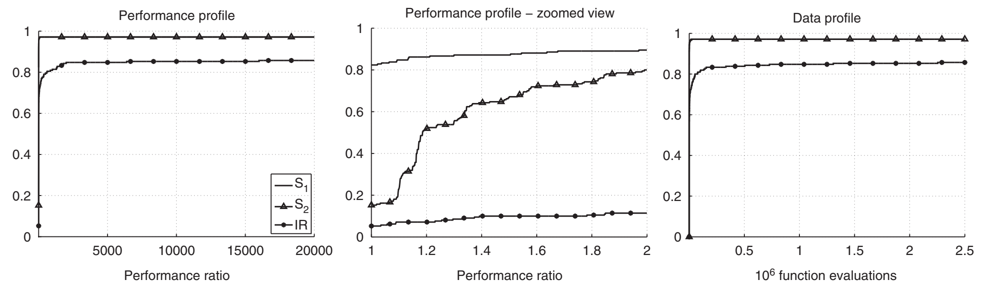

Figure 3. Performance and data profiles for the solvers S_1, S_2 and IR considering the precision 10^{-1} instead of 10^{-4} in (21).

where $f_{\min}$ is the smallest function value found among all the algorithms that are being compared. The criterion (21) will be used in the upcoming sections, though the values of $f_{\min}$ will be different, depending on which algorithms are being considered in each occasion.

The results are depicted in the performance and data profiles of Figure 2. Although the most robust algorithms were S_1 and S_2, the solver S_2 presented a slightly superior robustness. Algorithms S_1, S_2 and IR solved 92.9%, 95.2% and 70.5% of the problems, respectively. On the other hand, S_1, S_2 and IR were the most efficient for 79.5%, 17.6% and 5.7% of the problems, respectively.

Figure 3 presents the performance and data profiles replacing in (21) the value 10^{-4} by 10^{-1}, which is the precision adopted in [4]. Algorithms S_1 and S_2 solved 97.1% of the problems, while IR solved 85.7%. On the other hand, S_1, S_2 and IR were the most efficient for 82.4%, 15.2% and 5.2% of the problems, respectively.

4.3 *Comparison with solver* DFO

In this section, we compare the performance of algorithms S_1, S_2 and DFO in solving all 28 constrained problems of the Hock–Schittkowski collection [19] selected in [9]. The results for the DFO algorithm were extracted from [9]. We adopted (21) to decide if an algorithm solved a problem.

As shown in Figure 4, DFO was more efficient than our approach in this small set of problems. On the other hand, the three solvers were similar in terms of robustness, with S_1 and S_2 presenting a slightly superior performance. Algorithms S_1, S_2 and DFO solved 85.7%, 89.3% and 78.6% of the problems. In 5 of the 28 problems, the DFO algorithm terminated early because the algorithm used to solve the subproblems failed to produce a feasible solution [9].

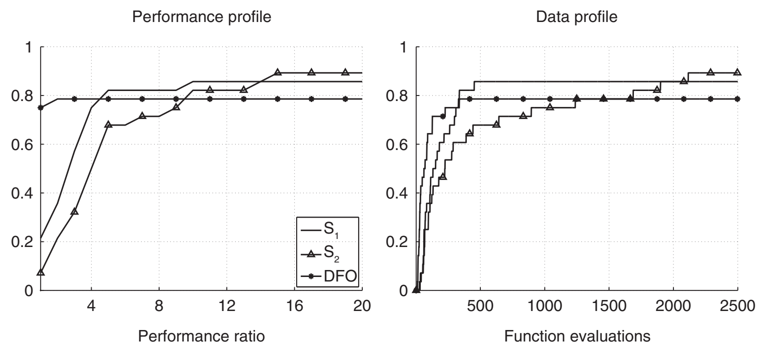

Figure 4. Performance and data profiles for the solvers S_1, S_2 and DFO.

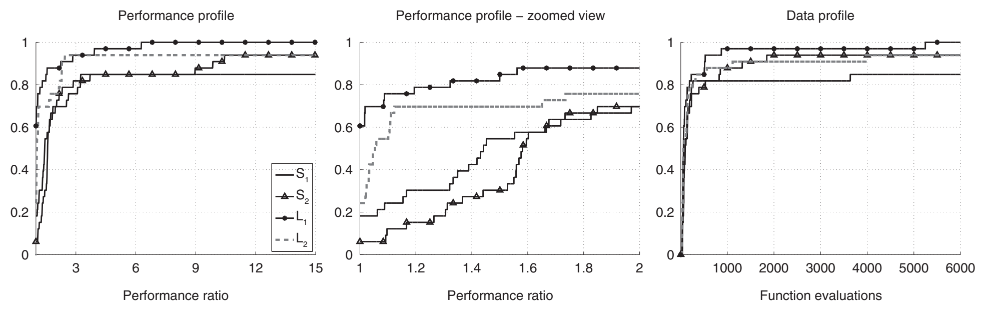

Figure 5. Performance and data profiles for the solvers S_1, S_2, L_1 and L_2.

4.4 *Comparison with solvers L_1 and L_2*

In this section, we present a numerical comparison with the LINCOA algorithm [22,28], which is an extension of BOBYQA [21,26] to treat linearly constrained optimization problems. As our algorithm and LINCOA are based on BOBYQA, they present some similarities. However, LINCOA is restricted to linearly constrained problems, while our algorithm treats problems with general constraints. Furthermore, LINCOA requires a feasible initial point. We compare the performance of S_1, S_2, L_1 and L_2 in solving all 33 linearly constrained optimization problems of the Hock–Schittkowski collection [19] in which the initial point is feasible, as required by the LINCOA software.

Figure 5 presents the performance and data profiles for number of function evaluations considering (21) to decide if an algorithm solved a problem. The solver L_1 was the most robust, solving all the problems, and also the most efficient, being the faster in terms of function evaluation for 60.6% of the problems. Algorithms L_2 and S_2 solved 93.9% of the problems, while S_1 solved 84.8%.

We present in Table 1 some more details on the numerical results. The first two columns contain information about the problems. The label prob indicates the number of the problem in the Hock–Schittkowski collection, and n, p, q the number of variables, of equality constraints and of inequality constraints besides bound constraints. The next columns display the minimum function value f^* and the number of function evaluations $\#f$ for each solver. Finally, the column f_{HS} displays the solution available in the Hock–Schittkowski collection.

When an algorithm failed in solving a problem, the following labels were used to indicate the reason:

Table 1. Numerical results.

		S₁			S₂			IR			DFO		L₁		L₂		
Prob.	*n*, *p*, *q*	f^*	#*f*		f^*	#*f*		f^*	#*f*		f^*	#*f*	f^*	#*f*	f^*	#*f*	f_{HS}
6	2, 1, 0	3.9229E − 26	36		6.2486E − 27	42		7.8394E − 10	370		–	–	–	–	–	–	1.8696E − 13
7	2, 1, 0	− 1.7321E + 00	30		− 1.7321E + 00	35		− 1.7321E + 00	154		–	–	–	–	–	–	− 1.7321E + 00
8	2, 2, 0	− 1.0000E + 00	26		− 1.0000E + 00	31		− 1.0000E + 00	4		–	–	–	–	–	–	− 1.0000E + 00
9	2, 1, 0	− 5.0000E − 01	40		− 5.0000E − 01	52		− 5.0000E − 01	118		–	–	− 5.0000E − 01	30	− 5.0000E − 01	31	− 5.0000E − 01
10	2, 0, 1	− 1.0000E + 00	28		− 1.0000E + 00	33		− 1.0000E + 00	259		–	–	–	–	–	–	− 1.0000E + 00
11	2, 0, 1	8.4985E + 00	27		− 8.4985E + 00	32		− 8.4985E + 00	126		–	–	–	–	–	–	− 8.4985E + 00
12	2, 0, 1	− 3.0000E + 01	42		− 3.0000E + 01	42		− 3.0000E + 01	588		–	–	–	–	–	–	− 3.0000E + 01
13	2, 0, 1	9.9913E − 01	36		9.9913E − 01	42		1.0886E + 00	46,392	(3)	–	–	–	–	–	–	1.0000E + 00
14	2, 1, 1	1.3935E + 00	26		1.3935E + 00	31		1.3935E + 00	20		–	–	–	–	–	–	1.3935E + 00
15	2, 0, 2	3.0650E + 00	30		3.0650E + 00	41		3.0650E + 00	82		–	–	–	–	–	–	3.0650E + 00
16	2, 0, 2	3.9821E + 00	34	(3)	3.9821E + 00	33	(3)	2.5000E − 01	671		–	–	–	–	–	–	2.3145E + 01
17	2, 0, 2	1.0000E + 00	1090		1.0000E + 00	94		1.0000E + 00	118		–	–	–	–	–	–	1.0000E + 00
18	2, 0, 2	5.0000E + 00	39		5.0000E + 00	45		5.0000E + 00	51,355		–	–	–	–	–	–	5.0000E + 00
19	2, 0, 2	− 6.9618E + 03	37		− 6.0524E + 03	58	(2,3)	− 6.9618E + 03	135		–	–	–	–	–	–	− 6.9618E + 03
20	2, 0, 3	3.8199E + 01	29		3.8199E + 01	34		4.0199E + 01	88	(3)	–	–	–	–	–	–	3.8199E + 01
21	2, 0, 1	− 9.9960E + 01	35		− 9.9960E + 01	41		− 9.9960E + 01	172		–	–	− 9.9960E + 01	30	− 9.9960E + 01	31	− 9.9960E + 01
22	2, 0, 2	1.0000E + 00	27		1.0000E + 00	32		1.0000E + 00	35		1.0000E + 00	15	–	–	–	–	1.0000E + 00
23	2, 0, 5	2.0000E + 00	35		2.0000E + 00	41		2.0000E + 00	37		2.0000E + 00	16	–	–	–	–	2.0000E + 00
24	2, 0, 3	− 1.0000E + 00	37		− 1.0000E + 00	43		− 1.0000E + 00	131		–	–	− 1.0000E + 00	34	− 1.0000E + 00	36	− 1.0000E + 00
26	3, 1, 0	3.1201E − 15	165		9.7794E − 15	177		1.7711E − 07	10,686		1.9355E − 09	49	–	–	–	–	7.4474E − 08
27	3, 1, 0	4.0000E + 00	107		4.0000E + 00	118		4.0003E + 00	29,184		–	–	–	–	–	–	4.0000E + 00
28	3, 1, 0	8.7449E − 21	61		2.6702E − 21	67		1.5820E − 24	524		–	–	1.2326E − 32	42	3.0815E − 33	43	3.0998E − 13
29	3, 0, 1	− 2.2627E + 01	93		− 2.2627E + 01	82		− 2.2627E + 01	571		–	–	–	–	–	–	− 2.2627E + 01
30	3, 0, 1	1.0000E + 00	59		1.0000E + 00	65		1.0000E + 00	788		–	–	–	–	–	–	1.0000E + 00
31	3, 0, 1	6.0000E + 00	58		6.0000E + 00	64		6.0000E + 00	536		–	–	–	–	–	–	6.0000E + 00
32	3, 1, 1	1.0000E + 00	63		1.0000E + 00	65		1.0000E + 00	84		1.0000E + 00	15	–	–	–	–	1.0000E + 00
33	3, 0, 2	− 4.0000E + 00	59		− 4.0000E + 00	65		− 4.0000E + 00	54		–	–	–	–	–	–	− 4.0000E + 00
34	3, 0, 2	− 8.3403E − 01	62		− 8.3403E − 01	68		− 8.3403E − 01	191		− 8.3403E − 01	22	–	–	–	–	− 8.3403E − 01
35	3, 0, 1	1.1111E − 01	50		1.1111E − 01	55		1.1111E − 01	475		–	–	1.1111E − 01	43	1.1111E − 01	36	1.1111E − 01
36	3, 0, 1	− 3.3000E + 03	62		− 3.3000E + 03	68		− 3.3000E + 03	257		–	–	− 3.3000E + 03	43	− 3.3000E + 03	45	− 3.3000E + 03
37	3, 0, 2	− 3.4560E + 03	191		− 3.4560E + 03	93		− 3.4560E + 03	474		–	–	− 3.4560E + 03	60	− 3.4560E + 03	59	− 3.4560E + 03
39	4, 2, 0	− 1.0000E + 00	58		− 1.0000E + 00	78		− 1.0000E + 00	435		–	–	–	–	–	–	− 1.0000E + 00
40	4, 3, 0	− 2.5000E − 01	60		− 2.5000E − 01	78		− 2.5000E − 01	127		–	–	–	–	–	–	− 2.5000E − 01

(*Continued*)

Table 1. Continued.

		S$_1$			S$_2$			IR			DFO			L$_1$		L$_2$		
Prob.	n, p, q	f^*	#f		f^*	#f		f^*	#f		f^*	#f		f^*	#f	f^*	#f	f_{HS}
41	4, 1, 0	1.9259E+00	170		1.9259E+00	81		1.9259E+00	456		–	–		–	–	–	–	1.9259E+00
42	4, 2, 0	1.3858E+01	71		1.3858E+01	95		1.3858E+01	536		–	–		–	–	–	–	1.3858E+01
43	4, 0, 3	−4.4000E+01	72		−4.4000E+01	96		−4.4000E+01	1506		–	–		–	–	–	–	−4.4000E+01
44	4, 0, 6	−1.3000E+01	96	(2-5)	−1.5000E+01	97		−1.3000E+01	264	(3)	−1.5000E+01	26		−1.5000E+01	45	−1.5000E+01	46	−1.5000E+01
46	5, 2, 0	3.4565E−13	249		4.1886E−13	381		5.0367E−07	5880		–	–		–	–	–	–	5.5456E−07
47	5, 3, 0	2.7152E−13	201		6.8727E−14	322		8.8559E−08	422		–	–		–	–	–	–	9.4223E−10
48	5, 2, 0	6.1937E−09	112		1.4175E−21	133		7.0719E−25	847		2.1132E−20	31		4.4184E−19	83	7.7037E−34	144	1.6289E−16
49	5, 2, 0	9.8752E−14	262		9.5195E−14	390		1.2096E−07	20,289		2.3516E−06	85		4.5713E−13	227	6.2985E−11	251	2.2836E−05
50	5, 3, 0	1.3090E−22	139		1.0989E−22	231		1.2599E−27	588		–	–		1.0182E−13	872	1.5797E−20	318	3.1211E−09
51	5, 3, 0	1.3014E−22	84		3.8998E−23	132		2.0094E−27	509		–	–		4.9304E−32	46	1.2326E−32	51	9.6183E−17
52	5, 3, 0	5.3266E+00	85		5.3266E+00	133		5.3267E+00	307		–	–		–	–	–	–	5.3266E+00
53	5, 3, 0	4.0930E+00	84		4.0930E+00	132		4.0930E+00	307		–	–		–	–	–	–	4.0930E+00
54	6, 1, 0	−9.0057E−01	1028	(3)	−9.0333E−01	2116		−1.5385E−01	444	(3)	−1.5391E−01	27	(1)	–	–	–	–	−7.2242E−34
55	6, 6, 0	6.6667E+00	76		6.6667E+00	141		6.6667E+00	18		–	–		–	–	–	–	6.6667E+00
56	7, 4, 0	−3.4560E+00	337		−3.4560E+00	224		−8.3922E−03	14,543	(3)	−3.4560E+00	37		–	–	–	–	−3.4560E+00
57	2, 0, 1	2.8460E−02	65		2.8460E−02	73		3.0646E−02	287	(3)	–	–		–	–	–	–	3.0646E−02
58	2, 0, 2	3.1903E+00	3005		3.1903E+00	75		3.1903E+00	103		–	–		–	–	–	–	3.1903E+00
59	2, 0, 3	−6.7546E+00	57	(3)	−6.7546E+00	57	(3)	−7.8042E+00	754		−6.7495E+00	32	(4)	–	–	–	–	−6.7546E+00
60	3, 1, 0	3.2568E−02	62		3.2568E−02	69		3.2568E−02	546		–	–		–	–	–	–	3.2568E−02
61	3, 2, 0	−1.4365E+02	61		−1.4365E+02	67		−1.4365E+02	180		–	–		–	–	–	–	−1.4365E+02
62	3, 1, 0	−2.6273E+04	65		−2.6273E+04	71		−2.6273E+04	811		–	–		−2.6273E+04	65	−2.6273E+04	73	−2.6273E+04
63	3, 2, 0	9.6172E+02	60		9.6172E+02	53		9.6172E+02	171		9.6172E+02	12		–	–	–	–	9.6172E+02
64	3, 0, 1	6.2998E+03	126		6.2998E+03	136		6.2998E+03	1438		–	–		–	–	–	–	6.2998E+03
65	3, 0, 1	9.5353E−01	62		9.5353E−01	68		9.5353E−01	1463		9.5353E−01	35		–	–	–	–	9.5353E−01
66	3, 0, 2	5.1816E−01	49		5.1816E−01	54		5.1816E−01	382		–	–		–	–	–	–	5.1816E−01
67	3, 0, 14	−1.1620E+03	132		−1.1620E+03	234		−1.1620E+03	25,632		–	–		–	–	–	–	−1.1620E+03
68	4, 2, 0	−9.2043E−01	310		−9.2043E−01	443		−9.2042E−01	5203		−9.2042E−01	127		–	–	–	–	−9.2043E−01
69	4, 2, 0	−9.5671E+02	330		4.0000E−03	246	(2,3,4)	−9.5671E+02	4125		−9.4134E+02	46	(1)	–	–	–	–	−9.5671E+02
70	4, 0, 1	7.4989E−03	4890		7.4985E−03	399		2.6909E−01	5571	(3)	–	–		–	–	–	–	7.4985E−03
71	4, 1, 1	1.7014E+01	74		1.7014E+01	80		1.7014E+01	4486		–	–		–	–	–	–	1.7014E+01
72	4, 0, 2	7.2768E+02	77	(3)	7.2148E+02	101		7.2768E+02	2724	(3)	–	–		–	–	–	–	7.2768E+02
73	4, 1, 2	2.9894E+01	59		2.9894E+01	79		2.9894E+01	211		–	–		–	–	–	–	2.9894E+01
74	4, 3, 2	5.1265E+03	83		5.1265E+03	123		5.1265E+03	392		5.1265E+03	71		–	–	–	–	5.1265E+03
75	4, 3, 2	5.1744E+03	78		5.1744E+03	103		5.1744E+03	139		–	–		–	–	–	–	5.1744E+03

(*Continued*)

Table 1. Continued.

		S_1			S_2			IR			DFO			L_1		L_2		
Prob.	n, p, q	f^*	#f		f^*	#f		f^*	#f		f^*	#f		f^*	#f	f^*	#f	f_{HS}
76	4, 0, 3	− 4.6818E + 00	73		− 4.6818E + 00	96		− 4.6818E + 00	483		− 4.6818E + 00	29		− 4.6818E + 00	51	− 4.6818E + 00	47	− 4.6818E + 00
77	5, 2, 0	2.4151E − 01	137		2.4151E − 01	205		2.4151E − 01	691		–	–		–	–	–	–	2.4151E − 01
78	5, 3, 0	− 2.9197E + 00	72		− 2.9197E + 00	108		− 2.9197E + 00	566		–	–		–	–	–	–	− 2.9197E + 00
79	5, 3, 0	7.8777E − 02	71		7.8777E − 02	109		7.8777E − 02	310		7.8777E − 02	25		–	–	–	–	7.8777E − 02
80	5, 3, 0	5.3950E − 02	73		5.3950E − 02	108		5.3950E − 02	644		–	–		–	–	–	–	5.3950E − 02
81	5, 3, 0	5.3950E − 02	144		5.3950E − 02	178		5.3950E − 02	774		–	–		–	–	–	–	5.3950E − 02
83	5, 0, 6	− 3.0666E + 04	87		− 3.0666E + 04	135		− 3.0666E + 04	430		–	–		–	–	–	–	− 3.0666E + 04
84	5, 0, 6	− 5.2803E + 01	104		− 5.2803E + 01	135		− 5.2803E + 01	350		–	–		–	–	–	–	− 5.2803E + 01
85	5, 0, 38	− 1.9052E + 00	107		− 1.1620E + 03	162		–	–		–	–		–	–	–	–	− 1.9052E + 00
86	5, 0, 10	− 3.2349E + 01	95		− 3.2349E + 01	132		− 3.2349E + 01	457		–	–		− 3.2349E + 01	62	− 3.2349E + 01	57	− 3.2349E + 01
87	6, 4, 0	8.9276E + 03	119	(3)	8.9276E + 03	210	(3)	8.8535E + 03	860		–	–		–	–	–	–	8.9276E + 03
88	2, 0, 1	1.3627E + 00	27		1.3627E + 00	32		1.3627E + 00	267		–	–		–	–	–	–	1.3627E + 00
89	3, 0, 1	1.3627E + 00	49		1.3627E + 00	54		1.3627E + 00	623		–	–		–	–	–	–	1.3627E + 00
90	4, 0, 1	1.3627E + 00	59		1.3627E + 00	79		1.3687E + 00	825	(3)	–	–		–	–	–	–	1.3627E + 00
91	5, 0, 1	1.3627E + 00	69		1.3627E + 00	109		1.3710E + 00	1743	(3)	–	–		–	–	–	–	1.3627E + 00
92	6, 0, 1	1.3627E + 00	94		1.3627E + 00	172		1.3659E + 00	1271	(3)	–	–		–	–	–	–	1.3627E + 00
93	6, 0, 2	1.3508E + 02	159		1.3508E + 02	178		1.3508E + 02	98,455		–	–		–	–	–	–	1.3508E + 02
95	6, 0, 4	1.5620E − 02	142		1.5620E − 02	77		1.5620E − 02	723		–	–		–	–	–	–	1.5620E − 02
96	6, 0, 4	1.5620E − 02	77		1.5620E − 02	142		1.5620E − 02	650		–	–		–	–	–	–	1.5620E − 02
97	6, 0, 4	3.1358E + 00	79		3.1358E + 00	144		4.0713E + 00	3763	(3)	–	–		–	–	–	–	3.1358E + 00
98	6, 0, 4	3.1358E + 00	79		3.1358E + 00	144		3.1358E + 00	1499		–	–		–	–	–	–	3.1358E + 00
99	7, 2, 0	− 8.3108E + 08	89		− 8.3108E + 08	183		− 8.3087E + 08	4	(3)	–	–		–	–	–	–	− 8.3108E + 08
100	7, 0, 4	6.8063E + 02	114		6.8063E + 02	226		6.8523E + 02	728,102	(3)	6.8063E + 02	127		–	–	–	–	6.8063E + 02
101	7, 0, 6	1.8098E + 03	940		1.8098E + 03	426		1.8098E + 03	464		–	–		–	–	–	–	1.8098E + 03
102	7, 0, 6	9.1188E + 02	241		9.1188E + 02	274		1.1364E + 03	48,584	(3)	–	–		–	–	–	–	9.1188E + 02
103	7, 0, 6	5.4367E + 02	411		5.4367E + 02	761		1.2644E + 03	384,575	(3)	–	–		–	–	–	–	5.4367E + 02
104	8, 0, 6	3.9512E + 00	153		3.9512E + 00	282		3.9512E + 00	8301		–	–		–	–	–	–	3.9512E + 00
105	8, 0, 1	1.1384E + 03	3637		1.1384E + 03	1846		1.1384E + 03	26,279		–	–		1.1384E + 03	5247	1.1384E + 03	3966	1.1384E + 03
106	8, 0, 6	7.0492E + 03	153		7.0492E + 03	289		7.0493E + 03	75,142		7.0492E + 03	63	(1)	–	–	–	–	7.0492E + 03
107	9, 6, 0	5.0550E + 03	108		5.0550E + 03	278		5.0550E + 03	265		5.0550E + 03	29		–	–	–	–	5.0550E + 03
108	9, 0, 13	− 8.6603E − 01	132		− 8.6603E − 01	896		− 4.9997E − 01	139,823	(3)	− 8.6603E − 01	62		–	–	–	–	− 8.6603E − 01
109	9, 6, 4	5.3621E + 03	141		5.3621E + 03	347		5.5949E + 03	2,254,623	(3)	–	–		–	–	–	–	5.3621E + 03

(*Continued*)

Table 1. Continued.

		S1			S2			IR			DFO			L1		L2		
Prob.	*n*, *p*, *q*	f^*	#*f*		f^*	#*f*		f^*	#*f*		f^*	#*f*		f^*	#*f*	f^*	#*f*	f_{HS}
111	10, 3, 0	− 4.7761E + 01	453		− 4.7761E + 01	1900		− 1.5978E + 35	2098	(1)	− 4.7761E + 01	227		–	–	–	–	− 4.7761E + 01
112	10, 3, 0	− 4.7571E + 01	150	(2–5)	− 4.7688E + 01	609	(2–5)	− 4.7761E + 01	4764		− 4.7761E + 01	337		− 4.7761E + 01	507	− 4.7761E + 01	1114	− 4.7761E + 01
113	10, 0, 8	2.4306E + 01	146		2.4306E + 01	404		2.4306E + 01	3403		–	–		–	–	–	–	2.4306E + 01
114	10, 3, 8	− 1.7688E + 03	182		− 1.7688E + 03	1688		− 1.7656E + 03	154,924	(3)	− 9.1628E + 02	8	(1)	–	–	–	–	− 1.7688E + 03
116	13, 0, 15	9.7588E + 01	218		9.7588E + 01	644		9.7589E + 01	33,965		9.7485E + 01	87	(1)	–	–	–	–	9.7591E + 01
117	15, 0, 5	3.2349E + 01	1927		3.2349E + 01	1248		3.2349E + 01	10,605		–	–		–	–	–	–	3.2349E + 01
118	15, 0, 29	6.6482E + 02	207		6.6482E + 02	825		6.6482E + 02	2010		–	–		6.6482E + 02	79	6.6482E + 02	182	6.6482E + 02
119	16, 8, 0	2.4490E + 02	299		2.4490E + 02	1238		2.4490E + 02	1200		2.4490E + 02	91		–	–	–	–	2.4490E + 02
215	2, 0, 1	− 1.2856E − 12	34	(3)	− 1.9052E + 00	40		− 5.3831E − 09	99	(3)	–	–		–	–	–	–	− 2.6592E − 09
216	2, 1, 0	9.9938E − 01	37		9.9938E − 01	43		9.9938E − 01	160		–	–		–	–	–	–	9.9938E − 01
217	2, 1, 1	− 8.0000E − 01	26		− 8.0000E − 01	31		− 8.0000E − 01	32		–	–		–	–	–	–	− 8.0000E − 01
218	2, 0, 1	0.0000E + 00	40		0.0000E + 00	46		4.9784E − 03	356	(3)	–	–		–	–	–	–	0.0000E + 00
219	4, 2, 0	− 1.0000E + 00	58		− 1.0000E + 00	78		− 1.0000E + 00	415		–	–		–	–	–	–	− 1.0000E + 00
220	2, 0, 1	1.0000E + 00	25		1.0000E + 00	30		3.0248E + 01	171	(3)	–	–		–	–	–	–	3.0240E + 01
221	2, 0, 1	− 1.0000E + 00	29		− 1.0000E + 00	34		− 9.5768E − 01	52,404	(3)	–	–		–	–	–	–	− 1.0000E + 00
222	2, 0, 1	− 1.5000E + 00	34		− 1.5000E + 00	40		− 1.5000E + 00	161		–	–		–	–	–	–	− 1.5000E + 00
223	2, 0, 2	− 8.3403E − 01	37		− 8.3403E − 01	43		− 8.3403E − 01	144		–	–		–	–	–	–	− 8.3403E − 01
224	2, 0, 4	− 3.0400E + 02	36		− 3.0400E + 02	42		− 3.0400E + 02	222		–	–		− 3.0400E + 02	36	− 3.0400E + 02	40	− 3.0400E + 02
225	2, 0, 5	2.0000E + 00	34		2.0000E + 00	40		2.0000E + 00	37		–	–		–	–	–	–	2.0000E + 00
226	2, 0, 2	− 5.0000E − 01	54		− 5.0000E − 01	33		− 5.0000E − 01	153		–	–		–	–	–	–	− 5.0000E − 01
227	2, 0, 2	1.0000E + 00	28		1.0000E + 00	33		1.0000E + 00	58		–	–		–	–	–	–	1.0000E + 00
228	2, 0, 2	− 3.0000E + 00	35		− 3.0000E + 00	41		− 3.0000E + 00	401		–	–		–	–	–	–	− 3.0000E + 00
230	2, 0, 2	3.7500E − 01	27		3.7500E − 01	32		3.7500E − 01	80		–	–		–	–	–	–	3.7500E − 01
231	2, 0, 2	1.9705E − 03	8384	(3)	1.6315E − 15	233		2.7605E − 06	24,277		–	–		2.5008E − 13	189	3.0698E − 20	126	6.5498E − 09
232	2, 0, 3	− 1.0000E + 00	34		− 1.0000E + 00	42		− 1.0000E + 00	126		–	–		− 1.0000E + 00	32	− 1.0000E + 00	33	− 1.0000E + 00
233	2, 0, 1	1.8725E − 05	83		4.8042E − 17	78		3.8501E − 05	45,591		–	–		–	–	–	–	4.5193E − 09
234	2, 0, 1	− 8.0000E − 01	92		− 8.0000E − 01	33		− 8.0000E − 01	3325		–	–		–	–	–	–	− 8.0000E − 01
235	3, 1, 0	4.0000E − 02	65		4.0000E − 02	65		4.0000E − 02	9874		–	–		–	–	–	–	4.0000E − 02
236	2, 0, 2	− 5.8903E + 01	40		− 5.8903E + 01	46		− 8.1975E + 00	333	(3)	–	–		–	–	–	–	− 5.8903E + 01
237	2, 0, 3	− 5.8903E + 01	53		− 5.8903E + 01	54		− 5.8901E + 01	18,569		–	–		–	–	–	–	− 5.8903E + 01
238	2, 0, 3	− 5.8903E + 01	40		− 5.8903E + 01	46		− 8.1975E + 00	629	(3)	–	–		–	–	–	–	− 5.8903E + 01
239	2, 0, 1	− 5.8903E + 01	40		− 5.8903E + 01	46		− 8.1975E + 00	349	(3)	–	–		–	–	–	–	− 5.8903E + 01
248	3, 1, 1	− 8.0000E − 01	59		− 8.0000E − 01	65		− 8.0000E − 01	321		–	–		–	–	–	–	− 8.0000E − 01

(*Continued*)

Table 1. Continued.

		S$_1$			S$_2$			IR			DFO		L$_1$		L$_2$		
Prob.	*n*, *p*, *q*	f^*	#*f*		f^*	#*f*		f^*	#*f*		f^*	#*f*	f^*	#*f*	f^*	#*f*	f_{HS}
249	3, 0, 1	1.0000E + 00	58		1.0000E + 00	64		1.0096E + 00	254	(3)	–	–	–	–	–	–	1.0000E + 00
250	3, 0, 2	– 3.3000E + 03	61		– 3.3000E + 03	67		– 3.3000E + 03	338		–	–	– 3.3000E + 03	43	– 3.3000E + 03	45	– 3.3000E + 03
251	3, 0, 1	– 3.4560E + 03	190		– 3.4560E + 03	92		– 3.4560E + 03	446		–	–	– 3.4560E + 03	60	– 3.4560E + 03	59	– 3.4560E + 03
252	3, 1, 0	4.0000E – 02	86		4.0000E – 02	95		4.0664E – 02	16,083	(3)	–	–	–	–	–	–	4.0000E – 02
253	3, 0, 1	6.9282E + 01	73		6.9282E + 01	80		6.9282E + 01	341		–	–	6.9282E + 01	51	6.9282E + 01	52	6.9282E + 01
254	3, 2, 0	– 1.7321E + 00	46		– 1.7321E + 00	51		– 1.7283E + 00	64	(3)	–	–	–	–	–	–	– 1.7321E + 00
262	4, 1, 3	– 1.0000E + 01	73		– 1.0000E + 01	97		– 1.0000E + 01	135		–	–	–	–	–	–	– 1.0000E + 01
263	4, 2, 2	– 1.0000E + 00	58		– 1.0000E + 00	78		– 9.7943E – 01	60	(3)	–	–	–	–	–	–	– 1.0000E + 00
264	4, 0, 3	– 4.4000E + 01	71		– 4.4000E + 01	95		– 4.2215E + 01	4895	(3)	–	–	–	–	–	–	– 4.4000E + 01
265	4, 2, 0	1.9036E + 00	55		1.9036E + 00	75		1.9036E + 00	160		–	–	–	–	–	–	1.9036E + 00
268	5, 0, 5	1.1159E + 00	2394	(3)	1.5092E + 00	132	(3)	1.9889E – 01	332,781		– 1.8190E – 11	84	3.9279E – 11	526	1.0232E – 12	134	1.3919E – 07
269	5, 3, 0	4.0930E + 00	83		4.0930E + 00	131		4.0930E + 00	307		–	–	–	–	–	–	4.0930E + 00
270	5, 0, 1	1.0639E – 10	101		1.7679E – 16	201		2.6695E – 11	453		–	–	–	–	–	–	– 1.0000E + 00
277	4, 0, 4	5.0762E + 00	59		5.0762E + 00	79		5.3196E + 00	638	(3)	–	–	–	–	–	–	5.0762E + 00
278	6, 0, 6	7.8385E + 00	96		7.8385E + 00	174		8.2349E + 00	3215	(3)	–	–	–	–	–	–	7.8385E + 00
279	8, 0, 8	1.0606E + 01	275		1.0606E + 01	275		1.0679E + 01	4910	(3)	–	–	–	–	–	–	1.0606E + 01
280	10, 0, 10	1.3375E + 01	143		1.3375E + 01	401		1.3375E + 01	142,946		–	–	–	–	–	–	1.3375E + 01
284	15, 0, 10	– 1.8400E + 03	202		– 1.8400E + 03	820		–	–		–	–	–	–	–	–	– 1.8400E + 03
285	15, 0, 10	– 8.2520E + 03	202		– 8.2520E + 03	820		–	–		–	–	–	–	–	–	– 8.2520E + 03
315	2, 0, 3	– 8.0000E – 01	35		– 8.0000E – 01	41		– 7.9853E – 01	159	(3)	–	–	–	–	–	–	– 8.0000E – 01
316	2, 1, 0	3.3431E + 02	38		3.3431E + 02	44		3.3432E + 02	156		–	–	–	–	–	–	3.3431E + 02
317	2, 1, 0	3.7247E + 02	37		3.7247E + 02	43		3.7247E + 02	151		–	–	–	–	–	–	3.7247E + 02
318	2, 1, 0	4.1275E + 02	38		4.1275E + 02	44		4.1275E + 02	151		–	–	–	–	–	–	4.1275E + 02
319	2, 1, 0	4.5240E + 02	37		4.5240E + 02	43		4.5240E + 02	201		–	–	–	–	–	–	4.5240E + 02
320	2, 1, 0	4.8553E + 02	37		4.8553E + 02	43		4.8553E + 02	128		–	–	–	–	–	–	4.8553E + 02
321	2, 1, 0	4.9611E + 02	37		4.9611E + 02	43		4.9611E + 02	240		–	–	–	–	–	–	4.9611E + 02
322	2, 1, 0	4.9996E + 02	37		4.9996E + 02	43		4.9996E + 02	219		–	–	–	–	–	–	4.9996E + 02
323	2, 0, 2	3.7989E + 00	28		3.7989E + 00	33		3.7989E + 00	184		–	–	–	–	–	–	3.7989E + 00
324	2, 0, 2	5.0000E + 00	38		5.0000E + 00	44		5.0000E + 00	51,244		–	–	–	–	–	–	5.0000E + 00
325	2, 1, 2	3.7913E + 00	35		3.7913E + 00	41		7.4641E + 00	42	(3)	–	–	–	–	–	–	3.7913E + 00
326	2, 0, 2	– 7.9808E + 01	34		– 7.9808E + 01	40		– 7.9808E + 01	101		–	–	–	–	–	–	– 7.9808E + 01
327	2, 0, 1	2.8460E – 02	71		2.8460E – 02	65		3.0646E – 02	287	(3)	–	–	–	–	–	–	3.0646E – 02
329	2, 0, 3	– 6.9618E + 03	37		– 6.9618E + 03	43		– 6.9618E + 03	120		–	–	–	–	–	–	– 6.9618E + 03

(*Continued*)

Table 1. Continued.

		S$_1$			S$_2$			IR			DFO		L$_1$		L$_2$		
Prob	*n, p, q*	f^*	#*f*		f^*	#*f*		f^*	#*f*		f^*	#*f*	f^*	#*f*	f^*	#*f*	f_{HS}
330	2, 0, 1	1.6206E + 00	62		1.6206E + 00	66		1.6206E + 00	245		–	–	–	–	–	–	1.6206E + 00
331	2, 0, 1	4.2584E + 00	62		4.2584E + 00	37		4.2584E + 00	283		–	–	4.2584E + 00	37	4.2584E + 00	39	4.2584E + 00
332	2, 0, 2	1.1495E + 02	29		1.1495E + 02	34		3.9268E + 01	1,012,052	(1)	–	–	–	–	–	–	1.1495E + 02
335	3, 2, 0	− 4.4721E − 03	46		− 4.4721E − 03	51		− 3.8311E − 01	6,904,369	(1)	–	–	–	–	–	–	− 4.4721E − 03
336	3, 2, 0	− 3.3790E − 01	48		− 3.3790E − 01	53		− 3.3790E − 01	253		–	–	–	–	–	–	− 3.3790E − 01
337	3, 0, 1	6.0000E + 00	58		6.0000E + 00	64		6.0000E + 00	519		–	–	–	–	–	–	6.0000E + 00
338	3, 2, 0	− 7.2057E + 00	59	(3)	− 7.2057E + 00	65	(3)	− 1.0993E + 01	6140		–	–	–	–	–	–	− 7.2057E + 00
339	3, 0, 1	3.3617E + 00	86		3.3617E + 00	83		3.3617E + 00	1150		–	–	–	–	–	–	3.3617E + 00
340	3, 0, 1	− 5.4000E − 02	151		− 5.4000E − 02	64		− 0.0000E + 00	20	(3)	–	–	− 5.4000E − 02	49	− 1.8165E − 01	57	− 5.4000E − 02
341	3, 0, 1	− 2.2627E + 01	81		− 2.2627E + 01	81		− 2.2627E + 01	658		–	–	–	–	–	–	− 2.2627E + 01
342	3, 0, 1	− 2.2627E + 01	92		− 2.2627E + 01	81		− 2.2627E + 01	658		–	–	–	–	–	–	− 2.2627E + 01
343	3, 0, 2	− 5.6848E + 00	69		− 5.6848E + 00	69		− 5.6848E + 00	157		–	–	–	–	–	–	− 5.6848E + 00
344	3, 1, 0	3.2568E − 02	61		3.2568E − 02	68		3.2568E − 02	635		–	–	–	–	–	–	3.2568E − 02
345	3, 1, 0	3.2568E − 02	67		3.2568E − 02	83		3.2595E − 02	103,325		–	–	–	–	–	–	3.2568E − 02
346	3, 0, 2	− 5.6848E + 00	69		− 5.6848E + 00	69		− 5.6848E + 00	157		–	–	–	–	–	–	− 5.6848E + 00
347	3, 1, 0	1.7375E + 04	62		1.7375E + 04	68		1.7375E + 04	287		–	–	1.7375E + 04	82	2.5698E + 04	34	1.7375E + 04
348	3, 1, 0	5.0689E + 01	45		5.0689E + 01	50		5.0689E + 01	2,521,555	(1)	–	–	–	–	–	–	3.6971E + 01
353	4, 1, 2	− 3.9934E + 01	56		− 3.9934E + 01	76		− 3.9934E + 01	500		–	–	–	–	–	–	− 3.9934E + 01
354	4, 0, 1	1.1378E − 01	144		1.1378E − 01	157		1.2756E − 01	415	(3)	–	–	1.1378E − 01	111	1.1378E − 01	109	1.1378E − 01
355	4, 1, 0	6.9675E + 01	120		6.9675E + 01	99		6.9676E + 01	36,407		–	–	–	–	–	–	1.2102E + 00
356	4, 0, 5	2.3812E + 00	84		2.3812E + 00	113		1.5815E + 01	9481	(3)	–	–	–	–	–	–	2.3812E + 00
359	5, 0, 14	− 5.2803E + 06	89		− 5.2803E + 06	138		− 3.2333E + 06	4	(3)	–	–	− 5.2803E + 06	139	− 5.2803E + 06	147	− 5.2803E + 06
360	5, 0, 2	− 5.2803E − 01	103		− 5.2803E − 01	134		− 2.5419E − 01	781	(3)	–	–	–	–	–	–	− 5.2803E − 01
361	5, 0, 6	− 7.7641E + 05	86		− 7.7641E + 05	134		− 7.7641E + 05	6		–	–	–	–	–	–	− 7.7641E + 05
362	5, 0, 4	2.6710E − 01	208		2.6760E − 01	363	(3)	2.0080E − 01	4,521,374	(1)	–	–	4.1700E − 02	138	4.1700E − 02	152	2.7080E − 01
364	6, 0, 4	6.1992E − 02	14366	(3)	6.0600E − 02	758		8.8959E − 02	4272	(3)	–	–	–	–	–	–	6.0600E − 02
365	7, 0, 5	2.3314E + 01	477		2.3314E + 01	221		0.0000E + 00	6,012,247	(1)	–	–	–	–	–	–	2.3314E + 01
366	7, 0, 14	7.0431E + 02	118		7.0431E + 02	231		− 6.6307E + 03	1,986,233	(1)	–	–	–	–	–	–	7.0431E + 02
367	7, 2, 3	− 3.7413E + 01	228		− 3.7413E + 01	223		− 3.7413E + 01	1161		–	–	–	–	–	–	− 3.7413E + 01
369	8, 0, 6	7.0492E + 03	131		7.0492E + 03	287		2.1000E + 03	6,577,668	(1)	–	–	–	–	–	–	7.0492E + 03
372	9, 0, 12	1.3390E + 04	140		1.3390E + 04	343		1.1001E + 05	2521	(3)	–	–	–	–	–	–	1.3390E + 04
373	9, 6, 0	1.3390E + 04	138		1.3390E + 04	345		1.3390E + 04	3200		–	–	–	–	–	–	1.3390E + 04
374	10, 0, 35	2.3326E − 01	142		2.3326E − 01	400		3.0438E − 01	262	(3)	–	–	–	–	–	–	2.3326E − 01

(*Continued*)

Table 1. Continued.

		S_1			S_2			IR			DFO		L_1		L_2		
Prob	n, p, q	f^*	$\#f$		f^*	$\#f$		f^*	$\#f$		f^*	$\#f$	f^*	$\#f$	f^*	$\#f$	f_{HS}
375	10, 9, 0	− 1.5161E + 01	144	(3)	− 1.5161E + 01	402	(3)	− 1.5624E + 01	20,161		–	–	–	–	–	–	− 1.5161E + 01
376	10, 1, 14	− 4.4301E + 03	284		− 4.4301E + 03	1028		− 3.0910E + 03	418,561	(3)	–	–	–	–	–	–	− 4.4301E + 03
377	10, 3, 0	− 7.9499E + 02	172		− 7.9499E + 02	475		− 7.9499E + 02	677		–	–	–	–	–	–	− 7.9499E + 02
378	10, 3, 0	− 4.7761E + 01	654		− 4.7761E + 01	2169		− 4.7753E + 01	1,307,311	(3)	–	–	–	–	–	–	− 4.7761E + 01
380	12, 0, 3	3.1687E + 05	291	(3)	3.1682E + 05	24,178		3.1687E + 05	504,668	(3)	–	–	–	–	–	–	3.2516E + 05
381	13, 1, 3	1.0149E + 00	149		1.0149E + 00	529		1.0149E + 00	2629		–	–	1.0149E + 00	59	1.0149E + 00	135	1.0149E + 00
382	13, 1, 3	1.0383E + 00	148		1.0383E + 00	528		1.0383E + 00	6273		–	–	–	–	–	–	1.0383E + 00
383	14, 1, 0	7.2859E + 00	229		7.2859E + 00	849		–	–		–	–	7.2859E + 00	513	7.2859E + 00	560	7.2859E + 00
384	15, 0, 10	8.3103E + 03	203		− 8.3103E + 03	821		–	–		–	–	–	–	–	–	− 8.3103E + 03
385	15, 0, 10	− 8.3153E + 03	203		− 8.3153E + 03	821		− 1.1282E + 03	1,275,790	(1)	–	–	–	–	–	–	− 8.3153E + 03
386	15, 0, 11	− 8.1644E + 03	203		− 8.1644E + 03	821		− 8.1416E + 03	1222	(3)	–	–	–	–	–	–	− 8.1644E + 03
387	15, 0, 11	− 8.2501E + 03	203		− 8.2501E + 03	821		− 8.2491E + 03	3064	(3)	–	–	–	–	–	–	− 8.2501E + 03
388	15, 0, 15	− 5.8211E + 03	204		− 5.8211E + 03	822		− 5.0809E + 03	86,477	(3)	–	–	–	–	–	–	− 5.8211E + 03
389	15, 0, 15	− 5.8097E + 03	204		− 5.8097E + 03	822		− 5.4083E + 03	78,056	(3)	–	–	–	–	–	–	− 5.8097E + 03
392	30, 0, 45	− 1.6961E + 06	805		− 1.6961E + 06	7463		–	–		–	–	–	–	–	–	− 1.6961E + 06
393	48, 2, 1	8.7393E − 01	1115	(3)	8.6338E − 01	12,276		1.5549E − 02	1,191,102	(1)	–	–	–	–	–	–	8.6338E − 01
394	20, 1, 0	1.9167E + 00	265		1.9167E + 00	1860		1.9167E + 00	37,025		–	–	–	–	–	–	1.9167E + 00
395	50, 1, 0	1.9167E + 00	625		1.9167E + 00	7963		1.9167E + 00	132,811		–	–	–	–	–	–	1.9167E + 00

(1) the problem was not solved because the solution is not feasible according to (19).
(2) the problem was not solved according to (20) considering solvers S_1 and S_2;
(3) the problem was not solved according to (21) considering solvers S_1, S_2 and `IR`;
(4) the problem was not solved according to (21) considering solvers S_1, S_2 and `DFO`;
(5) the problem was not solved according to (21) considering solvers S_1, S_2, L_1 and L_2.

5. Conclusion

In this work, we presented a trust-region algorithm for solving constrained optimization problems in which the derivatives of the objective function are not available. The algorithm was implemented in `Fortran` and its performance was compared with three solvers found in the literature. The numerical results are encouraging in terms of both efficiency and robustness. We believe that this good performance is a consequence of two facts. The first one is the usage of `ALGENCAN` [1,2] in Step 2, which proved to be very robust when solving the subproblems. The second one is that our approach extends to general-constrained problems the successful ideas proposed by Powell in [26] for box-constrained problems. The performance of a derivative-free trust-region algorithm depends immensely on the rules for choosing and updating the interpolation set and the quadratic model, and the ones proposed by Powell are among the best in the literature.

Acknowledgments

The authors are grateful to the anonymous referees, whose suggestions led to improvements in the paper. We also want to thank Francisco N. C. Sobral, for generously providing us the code for the Inexact Restoration algorithm [4].

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by CNPq, Brazil [grant number 477611/2013-3], [grant number 308957/2014-8].

Supplemental data and underlying research materials

The code of the algorithm and supplemental data for this article can be acessed at http://people.ufpr.br/ ~ ewkaras/pesquisa/publicacoes/supplemental_conejo_karas_pedroso.html.

References

[1] R. Andreani, E.G. Birgin, J.M. Martínez, and M.L. Schuverdt, *On augmented Lagrangian methods with general lower-level constraints*, SIAM J. Optimiz. 18 (2007), pp. 1286–1309.
[2] R. Andreani, E.G. Birgin, J.M. Martínez, and M.L. Schuverdt, *Augmented Lagrangian methods under the constant positive linear dependence constraint qualification*, Math. Program. 111 (2008), pp. 5–32.
[3] B.M. Arouxét, N. Echebest, and A. Pilotta, *Active-set strategy in Powell's method for optimization without derivatives*, Comput. Appl. Math. 30(1) (2011), pp. 171–196.
[4] L.F. Bueno, A. Friedlander, J.M. Martínez, and F.N.C. Sobral, *Inexact restoration method for derivative-free optimization with smooth constraints*, SIAM J. Optimiz. 23(2) (2013), pp. 1189–1213.
[5] P.G. Ciarlet and P.A. Raviart, *General Lagrange and Hermite interpolation in $\mathbb{R}^n$ with applications to finite elements methods*, Arch. Ration. Mech. Anal. 46 (1972), pp. 177–199.

[6] P.D. Conejo, E.W. Karas, L.G. Pedroso, A.A. Ribeiro, and M. Sachine, *Global convergence of trust-region algorithms for convex constrained minimization without derivatives*, Appl. Math. Comput.220 (2013), pp. 324–330.

[7] A.R. Conn, N.I.M. Gould, and Ph.L. Toint, *Trust-Region Methods*, MPS-SIAM Series on Optimization, SIAM, Philadelphia, 2000.

[8] A.R. Conn, K. Scheinberg, and Ph.L. Toint, *On the convergence of derivative-free methods for unconstrained optimization*, in *Approximation Theory and Optimization: Tributes to M.J.D. Powell*, M.D. Buhmann and A. Iserles, eds., Cambridge University Press, Cambridge, 1997, pp. 83–108.

[9] A.R. Conn, K. Scheinberg, and Ph.L. Toint, *A derivative free optimization algorithm in practice*, Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, CO, USA, September 1998.

[10] A.R. Conn, K. Scheinberg, and L.N. Vicente, *Geometry of interpolation sets in derivative free optimization*, Math. Program. 111 (2008), pp. 141–172.

[11] A.R. Conn, K. Scheinberg, and L.N. Vicente, *Geometry of sample sets in derivative-free optimization: Polynomial regression and undetermined interpolation*, IMA J. Numer. Anal. 28(4) (2008), pp. 721–748.

[12] A.R. Conn, K. Scheinberg, and L.N. Vicente, *Global convergence of general derivative-free trust-region algorithms to first and second order critical points*, SIAM J. Optimiz. 20(1) (2009), pp. 387–415.

[13] A.R. Conn, K. Scheinberg, and L.N. Vicente, *Introduction to Derivative-Free Optimization*, MPS-SIAM Series on Optimization, SIAM, Philadelphia, 2009.

[14] A.R. Conn and Ph.L. Toint, *An Algorithm using Quadratic Interpolation for Unconstrained Derivative Free Optimization*, in *Nonlinear Optimization and Applications*, G. Di Pillo and F. Gianessi, eds., Plenum Press, New York, 1996, pp. 27–47.

[15] E.D. Dolan and J.J. Moré, *Benchmarking optimization software with performance profiles*, Math. Program. 91(2) (2009), pp. 201–213.

[16] G. Fasano, J.L. Morales, and J. Nocedal, *On the geometry phase in model-based algorithms for derivative-free optimization*, Optim. Method. Softw. 24 (2009), pp. 145–154.

[17] P.E. Gill, W. Murray, M.A. Saunders, and M.H. Wright, *User's guide for NPSOL (version 4.0): A Fortran package for nonlinear programming*. Technical Report SOL 86–2, Department of Operations Research, Stanford University, USA, 1986.

[18] S. Gratton, P.L. Toint, and A. Tröltzsch, *An active set trust-region method for derivative-free nonlinear bound-constrained optimization*, Optim. Method. Softw. 26 (2011), pp. 873–894.

[19] W. Hock and K. Schittkowski, *Test examples for nonlinear programming codes*, Lecture Notes in Economics and Mathematical Systems, Vol. 187, 1981.

[20] J.J. Moré and S.M. Wild, *Benchmarking derivative-free optimization algorithms*, SIAM J. Optimiz.20(1) (2009), pp. 172–191.

[21] M.J.D. Powell, *BOBYQA software*. Software available at http://mat.uc.pt/ ~ zhang/software.html#bobyqa.

[22] M.J.D. Powell, *LINCOA software*. Software available at http://mat.uc.pt/ ~ zhang/software.html#lincoa.

[23] M.J.D. Powell, *Least Frobenius norm updating of quadratic models that satisfy interpolation conditions*, Math. Program. 100 (2004), pp. 183–215.

[24] M.J.D. Powell, *The NEWUOA software for unconstrained optimization without derivatives*, in *Large-Scale Nonlinear Optimization*, G. Di Pillo and M. Roma, eds., Springer, New York, 2006, pp. 255–297.

[25] M.J.D. Powell, *Developments of NEWUOA for minimization without derivatives*, IMA J. Numer. Anal. 28 (2008), pp. 649–664.

[26] M.J.D. Powell, *The BOBYQA algorithm for bound constrained optimization without derivatives*, Technical Report DAMTP NA2009/06, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, UK, August 2009.

[27] M.J.D. Powell, *On the convergence of trust region algorithms for unconstrained minimization without derivatives*, Comput. Optim. Appl. 53(2) (2012), pp. 527–555.

[28] M.J.D. Powell, *On fast trust region methods for quadratic models with linear constraints*, Technical Report DAMTP NA2014/02, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, UK, August 2014.

[29] K. Scheinberg and Ph.L. Toint, *Self-correcting geometry in model-based algorithms for derivative-free unconstrained optimization*, SIAM J. Optimiz. 20(6) (2010), pp. 3512–3532.

[30] K. Schittkowski, *NLPQLP: A Fortran implementation of a sequential quadratic programming algorithm with distributed and non-monotone line search - user's guide, version 2.2*, Technical report, Department of Computer Science, University of Bayreuth, Bayreuth, Germany, 2006.

[31] K. Schittkowski, *An updated set of 306 test problems for nonlinear programming with validated optimal solutions – user's guide*, Technical report, Department of Computer Science, University of Bayreuth, Bayreuth, Germany, 2008.