A Derivative-Free and Ready-to-Use NLP Solver for Matlab or Octave

Florian Jarre and Felix Lieder,
Mathematisches Institut,
Heinrich-Heine Universität Düsseldorf, Germany.

Abstract

This paper introduces a derivative-free and ready-to-use solver for nonlinear programs with nonlinear equality and inequality constraints (NLPs). Using finite differences and a sequential quadratic programming (SQP) approach, the algorithm aims at finding a local minimizer and no extra attempt is made to generate a globally optimal solution. Due to the use of finite differences, approximations of the derivatives are expensive compared to the numerical computations that usually dominate the computational effort of NLP solvers. This fact motivates the use of a somewhat effortful trust-region SQP-subproblem that is solved by second orde cone programs.

The implementation in Matlab or Octave is easy to use and public domain; numerical experiments indicate that the algorithm is well suitable for problems with m inequality constraints depending on n variables when $n + m \leq 500$.

Key words: Minimization without derivatives, nonlinear programs, sequential quadratic programming solver, Matlab.

May 4, 2017

1 Introduction

An iterative algorithm is presented for minimizing a smooth function subject to smooth equality- and inequality-constraints, but without using any user supplied derivative information. Instead, the derivatives are approximated via central finite differences. The finite difference approximations are expensive compared to computations of numerical linear algebra which tend to dominate the overall computational cost in standard NLP solvers. The consideration of this fact motivates a conforming strategy for the numerical computation of the iterates.

The concepts used in this paper are fairly standard and there are many alternative standard approaches that could be used instead. The new contribution of the present paper is to apply and implement these concepts to an algorithm

without using derivatives and to combine them in a way which merits the fact that approximations to derivatives are moderately expensive.

For problems with very expensive function evaluations other approaches are preferable, for example the recent DEFT-FUNNEL approach by Sampaio and Toint [13].

In spite of the fact that no derivative information is provided by the user, the numerical experiments in [9] indicate that using curvature approximations may significantly accelerate the convergence of a descent method. For this reason, in the implementation [7] associated with the present paper, a rather high numerical effort is spent on generating a suitable projection of an approximation to the Hessian of the Lagrangian. This projection is used in a Euclidean norm trust region SQP subproblem proposed by Celis, Dennis, and Tapia, [3].

Note that – due to the finite difference approach – this algorithm is not suitable for large scale problems, in particular, sparsity is not exploited.

1.1 Notation

The components of a vector $x \in \mathbb{R}^n$ are denoted by x_i for $1 \leq i \leq n$. When $x \in \mathbb{R}^n$ is some vector, $\|x\|$ denotes its Euclidean norm. For $lb, ub \in (\mathbb{R} \cup \{\pm\infty\})^n$, inequalities such as $lb \leq x \leq ub$ are understood componentwise.

The gradient of a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ at some point x will be denoted by a column vector $g(x) = \nabla f(x) = Df(x)^T$, and the Hessian (if it exists) by $H(x) = \nabla^2 f(x)$.

For a vector $y \in \mathbb{R}^m$ the vector y_+ denotes the vector with entries

$$(y_+)_i := \max\{0, y_i\} \qquad \text{for } \; 1 \leq i \leq m.$$

The k-th iterate of a sequence of vectors is denoted by a superscript, such as x^k; for sequences of scalars or matrices, the k-th element is denoted by a subscript such as δ_k or H_k.

The machine precision is denoted by ϵ.

2 The problem

The problem under consideration is of the general format

$$\begin{array}{lcll} \text{minimize} & f(x) & | & \begin{array}{ll} f_{E_1}(x) = 0, & A_{E_2} x = b_{E_2}, \\ f_{I_1}(x) \leq 0, & A_{I_2} x \leq b_{I_2}, \quad lb \leq x \leq ub, \end{array} \end{array} \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $f_{E_1} : \mathbb{R}^n \to \mathbb{R}^p$, $f_{I_1} : \mathbb{R}^n \to \mathbb{R}^q$ are differentiable functions, and A_{E_2}, A_{I_2} are real matrices with n columns. The dimensions of the right hand side "0" and of b_{E_2}, b_{I_2} above are to conform with the definitions of $f_{E_1}, f_{I_1}, A_{E_2}, A_{I_2}$. In particular, "0" may be the empty zero-vector if there are no nonlinear equality constraints (E_1 is empty) or no nonlinear inequality constraints (I_1 is empty), and likewise b_{E_2} or b_{I_2} may have zero rows if A_{E_2} or A_{I_2} do so. The lower and upper bound constraints on x can also be omitted, or some components of lb may be "$-Inf$" or some components of ub may be "Inf". It is assumed that a finite initial point x^0 is given where x^0 does not necessarily satisfy the constraints.

The constraint function may or may not depend on additional parameters that are not subject to minimization. The input format is detailed next.

2.1 The use in Matlab/Octave, input and output format

To facilitate the use of the minimization routine, only a single m-file "`min_fc.m`" needs to be copied into the folder from which Matlab or Octave are called (or into some other folder on the matlab-path).

Using Matlab-notation, the problem to be solved by "`min_fc.m`" (**min**imization of a function **f** subject to general **c**onstraints) is

```
minimize  f_obj(x)  for  f_e(x)  = 0,   f_i(x) <= 0,
                         Ae*x     = be,  Ai*x   <= bi,
                                lb <= x <= ub
```

The general calling routine of "`min_fc`" is of the form

```
[x,y,fx,out] = min_fc(@f_obj,@f_con,options)
```

where the output for the constraint function f_con consists of two vectors f_e (equality constraints) and f_i (inequality constraints). The objective and constraints may depend on additional parameters (not subject to minimization) as detailed below.

The input data is structured as follows:

```
Mandatory input:
    f_obj, a function handle for the objective function R^n --> R
    f_con, a function handle for the constraints R^n --> [R^p, R^q]
!!!  Even when p=0 or q = 0, the output of f_con must consist of two   !!!
!!!  vectors [fe ,fi] = f_con(x), with fe  = zeros(0,1) if p = 0,      !!!
!!!  or fi = zeros(0,1) if q = 0.  Here, p and q do not have to be     !!!
!!!  specified; they will be determined from fe and fi.                !!!
!!!  (fe and fi must be column vectors.)                               !!!
------------------------------------------------------------------------
Further MANDATORY input:
    options, a structure with the MANDATORY field
    x0, starting point, not necessarily feasible but of correct dimension
------------------------------------------------------------------------
  and the further OPTIONAL fields:
    Ae, a constraint matrix for linear equality constraints
    be, right hand side  (default for Ae, be:  empty)
    Ai, a constraint matrix for linear inequality constraints
    bi, right hand side  (default for Ai, bi:  empty)
    lb, lower bounds on the variable x, of dim (n,1)   (Default -Inf)
    ub, upper bounds on the variable x, of dim (n,1)   (Default  Inf)
        (Sparsity of the bounds is not exploited.  Setting lb_i=-Inf or
         ub_i=Inf is more efficient than e.g., lb_i=-1e20 or ub_i=1e20.)
    par_fobj, If f_obj depends on additional parameters (not subject
              to optimization), then the field options.par_fobj is a
              struct containing the input parameters.
              Calling routine:  f_obj(x,par_fobj)
              Default:  options.par_fobj is not provided
    par_fcon, If f_con depends on additional parameters (not subject
              to optimization), then the field options.par_fcon is a
              struct containing the input parameters.
              Default:  options.par_fcon is not provided
    maxit, a bound on the number of iterations; each iteration
           takes about 2*n+15 evaluations of f_obj, f_con (Default 100*n)
    err,   if the absolute error for a typical evaluation of f_obj or
           f_con is known, this parameter can be set here, else it
           is estimated (and used for the finite differences).
```

```
  p_l,   print_level, values 1 or 2 for more printout, default 0
```

The output is given as follows:

```
x:        an approximate local minimizer / stationary point
y:        the associated Lagrange multipliers in the order:  fe, Ae,
          fi, Ai, (Ai including the bounds lb,ub)
          y is empty if there is only one degree of freedom (after
          the elimination of linear constraints) i.e.  if the problem
          can be reduced to a line search
fx    the final objective value
out.relKKTres:  Norm of the gradient of the Lagrangian divided by
                max(1,norm of the gradient of the objective function)
out.constr_viol = [norm(constraint violation, nonlinear   equalities),
                   norm(constraint violation,    linear   equalities),
                   norm(constraint violation, nonlinear inequalities),
                   norm(constraint violation,    linear inequalities)];
out.iter:  number of iterations needed
out.fval:  number of function evaluations needed
out.Ae     transformed matrix of linear   equalities used for ye
out.Ain    transformed matrix of linear inequalities used for yin
out.termmsg termination message
            1) KKT point found with 6 digits accuracy
            2) infeasible stationary point discovered with 6 digits acc
            3) KKT point found with 3 digits accuracy
            4) Early termination as iteration limit has been reached
            5) Termination without reaching convergence
            6) Termination without improving over the initial point
```

Termination message 5) may occur if the current SQP step does not result in a sufficient decrease of the merit function; termination message 6) indicates that this situation is given at the initial point. The termination messages are based on estimates and are not reliable.

Below we list a sample file for calling `min_fc`. The objective and the constraints are given by the following functions:

```
function [ y ] = f_sample_obj( x,A )
% f_sample_obj, a simple quadratic objective function
% depending on a parameter matrix A not subject to minimization
y = x'*A*x;
end


function [ feq, fin ] = f_sample_con( x )
feq = x(1)^2; % degenerate representation of the constraint x(1) = 0
fin = [x(2)+x(3)-0.8;
       exp(x(3))-1-x(4)];
% a linear inequality constraint but treated like a nonlinear inequality
% followed by the convex constraint x(4) >= exp(x(3))-1
end
```

To prepare the call set, for example:

```
options.Ae = ones(1,4);  % linear equality constraint sum(x_i) = 1
options.be = 1;
options.lb = zeros(4,1); % lower bounds all zero, no upper bounds
A = [ 6, -2, -3, -4; -2, 9, 1, 2; -3, 1, -3, -3; -4, 2, -3, -1];
options.par_fobj = A;    % parameter matrix for the objective function
options.x0 = ones(n,1);  % some (infeasible) starting point
options.p_l = 1;         % moderate printing level
```

The actual function call:

```
[x,y,fx,out] = min_fc(@f_sample_obj,@f_sample_con,options);
```

The output for this particular example is:

```
x =
  0.000000019116584
  0.000000004849712
  0.442854389555730
  0.557145586477974


y =
  1.0e+09 *
  8.344948992241799
 -0.000000009285859
                   0
  0.000000000871512
  0.000000316582474
  0.000000007757220
                   0
                   0


out =
 struct with fields:


       termmsg:  1
     relKKTres:  4.4061e-08
   constr_viol:  [3.6544e-16 1.1102e-16 0 0]
          iter:  17
          fval:  437
           Ain:  [44 double]
            Ae:  [-0.5000 -0.5000 -0.5000 -0.5000]
           bin:  [41 double]
            be:  -0.5000
```

The vector `y` of Lagrange multipliers first lists the multipliers for the nonlinear and the linear equality constraints – here, the degeneracy of the problem results in a huge multiplier for the nonlinear equality constraint – and then for the nonlinear and the linear inequality constraints (in this order).

The struct `out` lists the following results: The termination message 1 indicates that the constraint violation and the estimated relative residual of the KKT violation are less than 10^{-6}. Here, the relative residual is the residual divided by "the norm of the objective gradient +1". The constraint violation `constr_viol` is listed in the order "nonlinear equality constraints, linear equality constraints, nonlinear inequality constraints, linear inequality constraints". The number of iterations and the number of function evaluations follow. The matrix `Ain` represents the linear inequalities – here four inequalities coming from the lower bounds. The matrix `Ae` is from the linear equality constraint; this matrix is always rescaled to be a matrix with orthonormal rows. The vector `bin` is the right hand side for the linear inequality constraints[1] and `be` is the right hand side for the linear equality constraint(s).

[1]The algorithm proceeds by first eliminating the linear equations, which results in modifying also the inequalities. When converting the output back to the original form, the rows

3 A nonlinear minimization principle

The algorithm used in "`min_fc.m`" aims at finding a local minimizer near x^0 but in general, it will not maintain exact feasibility with respect to the nonlinear constraints, even if the initial iterate does satisfy all constraints.

The SQP-penalty-approach detailed in sections 3.1 – 3.3.1 and implemented in [7] follows well known principles: The great success of SQP methods is owed to the fact that it is possible to collect all relevant second order information in a single matrix, the Hessian of the Lagrangian, rather than generating second order information for each individual constraint function. This comes at the expense that the accuracy of an SQP step cannot be measured easily, resulting in a possible Maratos-effect [10]. Here, it is attempted to suppress this effect by considering convex sub-problems with a descent property and by including second order correction steps.

3.1 Preprocessing

The algorithm starts by projecting x^0 onto the nearest point satisfying the linear equations and inequalities. Then, the equations $A_{E_2}x = b_{E_2}$ are eliminated[2] using a QR-factorization of A_{E_2}, and the finite bounds lb_j, ub_j $(1 \le j \le n)$ are included in A_{I_2}, b_{I_2}.

The result of this elimination[3] is a problem of the format

$$\text{minimize} \quad f(x) \quad | \quad f_E(x) = 0, \quad f_I(x) \le 0, \quad Ax \le b. \tag{2}$$

Here, the definitions of f, f_E, f_I and A, b have changed compared to (1); in particular, if E_2 is nonempty, then, after eliminating A_{E_2}, b_{E_2}, the dimension of the input to f, f_E, f_I is smaller than in problem (1). This implies that also the starting point x^0 is projected to a point in a lower dimensional space satisfying the linear equalities and inequalities. A linear inequality constraint that is satisfied as an equality for all points satisfying all linear constraints is called a hidden equality. It is assumed that the problem contains no hidden linear equality constraints. The set of rows of A in (2) is denoted by J and the Lagrange multipliers for the three types of constraints in (2) are denoted by y_E, y_I, and y_J: The Lagrangian is given by

$$L(x, y_E, y_I, y_J) := f(x) + y_E^T f_E(x) + y_I^T f_I(x) + y_J^T(Ax - b).$$

The input in the implementation [7] is of the format (1). Then, the above preprocessing is applied followed by a call to the actual minimization algorithm. For the theoretical description of the algorithm we may therefore assume without loss of generality (and without deviating from the notation used within the implementation [7]) that the problem is given in the format (2) with a starting point that satisfies all linear constraints. The feasibility with respect to

of `Ain` and the entries of `bin` may be modified by multiples of the equality constraints. In the above example `Ain` is no longer a multiple of the identity matrix – but for any `x` with `Ae * x = be` it still represents the inequalities `x >= 0`.

[2] When A_{E_2} is not the empty matrix (with zero rows), such elimination typically destroys the sparsity that may be present in A_{I_2}. As pointed out above, sparsity is not exploited in [7].

[3] The elimination is carried out internally and is not visible to the user.

the linear constraints will be maintained at all steps of the minimization process; therefore, in the following, only the constraint violation of the nonlinear constraints f_E, f_I will be considered.

3.1.1 Initialization

As initialization set the iteration index $k = 0$ and compute the function values $f^k = f(x^k)$, $f_E^k = f_E(x^k)$, $f_I^k = f_I(x^k)$, and finite difference approximations $Df^k \approx Df(x^k)$, $Df_E^k \approx Df_E(x^k)$, and $Df_I^k \approx Df_I(x^k)$ based on central finite differences. As in [9], the size "dt" of the discretization for the finite difference is adapted to an estimate of the accuracy of the function values. Apart from the high computational effort involved with the finite difference evaluations, also a loss of accuracy is associated with Df^k, Df_E^k and Df_I^k compared to an evaluation of an analytic expression for the derivatives. Nevertheless, for notational convenience, the approximation of the derivatives by finite differences will be denoted shortly as "evaluation of the derivatives" below.

The algorithm will also use an approximation H_k of the Hessian of the Lagrangian at the k-th iteration. Initially, the Hessian is set to $H_0 = 0$.

3.2 SQP concept

The algorithm of Section 3.3 generates a sequence of iterates x^k where x^{k+1} is obtained from the solution of a convex trust region SQP subproblem at the point x^k. As detailed in [8], the SQP subproblem proposed by Celis, Dennis, and Tapia [3] for equality constrained problems is generalized to equality- and inequality-constrained problems and used in the implementation of Section 3.3. This subproblem is termed CDT subproblem below. The CDT subproblems allow a strong convergence analysis in [3] but in relying on second-order cone programs they are computationally more expensive than other formulations of SQP subproblems. Due to the high computational effort for generating derivative approximations, the relative increase of the overall computational cost due to solving the CDT subproblem, however, is moderate, in general.

3.2.1 Projection of the Hessian

Given an iterate x^k, the function values f^k, f_E^k, f_I^k, the finite difference approximations Df^k, Df_E^k, Df_I^k, and some approximation $H_k \approx \nabla^2 L(x^k, y_E^k, y_I^k, y_J^k)$, a trust region SQP subproblem will be set up. To this end H_k is projected onto the space orthogonal to the gradients of the active constraints. Initially, at iteration $k = 0$, only the equality constraints that are considered active. For the iterations $k \geq 1$, all linear and nonlinear inequalites that are violated or whose slacks are less than twice the norm of the second order correction of the previous iteration are considered "active" as well.

The (possibly negative) curvature perpendicular to the active constraints will be eliminated by projecting the Hessian H_k onto the subspace orthogonal to Df_E^k and orthogonal to the derivatives of all active constraints. More precisely, let $\tilde{A}$ be the matrix consisting of Df_E^k and of the active rows of Df_I^k and the active rows of A, i.e.

$$\tilde{A} := \begin{bmatrix} Df_E^k \\ (Df_I)_{act}^k \\ A_{act} \end{bmatrix}.$$

Let $\tilde{A}^T = QR$ be a QR-decomposition, where $Q = [Q_1, Q_2]$ and Q_2 is associated with the zero-rows of R. Then set $\tilde{H}_k := Q_2 Q_2^T H_k Q_2 Q_2^T$. (If Q_2 is empty then $\tilde{H}_k$ is set to zero.) The matrix $\tilde{H}_k$ is called *projected Hessian* and is closely related to the reduced Hessian[4], see for example [1, 2, 6, 11]. If the smallest eigenvalue of $\tilde{H}_k$ satisfies $\lambda_{min}(\tilde{H}_k) \leq 0$, then a further correction is applied to $\tilde{H}_k$ by adding a regularization term and changing $\tilde{H}_k$ to $\tilde{H}_k := \tilde{H}_k + \rho I$ where $\rho \geq -\lambda_{min}(\tilde{H}_k)$. Below, it is assumed that $\tilde{H}_k \succeq 0$.

3.3 The CDT SQP step

Given x^k, $\tilde{H} = \tilde{H}_k$ and a trust region radius $\delta > 0$, let $g = g^k := (Df^k)^T$, let $Df_E = Df_E^k$ be the (approximate) Jacobian of the equality constraints evaluated at x^k, and let $f_E = f_E^k$ be the function value of the equality constraints evaluated at x^k. Likewise, let Df_I and f_I be approximations for the Jacobian and function value of the inequality constraints.

Following the outline in [8] a correction step Δx is determined by the solution of the following two problems: First, let $\rho \in (0, 1)$ be some fixed number, e.g. $\rho = 0.9$, and let $t_k = \delta_C$ be the optimal value of

$$\begin{array}{lrcccl} \min & & & t & & \\ & Df_E\Delta\tilde{x} & - & s_E & = & -f_E, \\ & Df_I\Delta\tilde{x} & - & s_I & \leq & -f_I, \\ \text{s.t.} & & & \left\| \begin{pmatrix} s_E \\ s_I \end{pmatrix} \right\| & \leq & t, \\ & & & \|\Delta\tilde{x}\| & \leq & \rho\ \delta. \end{array} \tag{3}$$

Then, the CDT correction Δx is given by the solution of

$$\begin{array}{lrcccl} \min & & & g^T\Delta x + \frac{1}{2}\Delta x^T \tilde{H} \Delta x & & \\ & Df_E\Delta x & - & s_E & = & -f_E, \\ & Df_I\Delta x & - & s_I & \leq & -f_I, \\ \text{s.t.} & & & \left\| \begin{pmatrix} s_E \\ s_I \end{pmatrix} \right\| & \leq & \delta_C, \\ & & & \|\Delta x\| & \leq & \delta. \end{array} \tag{4}$$

Problem (3) can trivially be rewritten as a mixed second order cone program, and by the preprocessing of $\tilde{H}$ it follows that $\tilde{H} \succeq 0$ so that also problem (4) can be rewritten as a mixed second order cone program, the solution of which is discussed, e.g. in [8].

[4]Several other approaches for modifying the Hessian have been proposed. For example, instead of the above projection, in order to eliminate negative curvature orthogonal to the active constraints, it is also possible to define $\tilde{H}_k := H_k + \rho\tilde{A}^T\tilde{A}$ for sufficiently large $\rho \geq 0$. As the appropriate choice of ρ is not known, the above projection is applied instead.

Projecting the Hessian H_k to $\tilde{H}_k$ as above or leaving it unchanged will generate exactly the same steps when the linearizations of active constraints are fixed as equations. This is the case in active set methods, where, rather than a projected Hessian, a reduced Hessian is typically used instead. However, as long as the active constraints at optimality are not correctly identified, the active set may limit the step length to a rather short step even in cases, where the current iterate is far from a local minimizer and where the quadratic model at the current iterate might be sufficiently accurate in a much larger trust region. For this reason an SQP type approach is chosen below, avoiding the restriction to the "currently active set".

As output for problem (4), the algorithm in [8] returns an approximate value Δx^k.

Due to the trust region constraint, the Lagrange multipliers of (4) may not be good approximations to the Lagrange multipliers of (2). Approximate values for the Lagrange multipliers for (2) can be determined by the solution of a further second order cone program.

To avoid or to reduce a possible Maratos-effect [10] and to speed up overall convergence associated with a single evaluation of all constraint gradients, a second order correction step s^k is computed for the point $x^k + \Delta x^k$: To this end let $\hat{I}^+$ denote the inequalities violated at $x^k + \Delta x^k$, let $\hat{b}^+$ denote the right hand side associated with the remaining inequalites, and let $\hat{A}^+$ be defined conforming with $\hat{b}^+$. Further, let $f_E^+ := f_E(x^k + \Delta x^k)$ and $f_{\hat{I}^+}^+ := f_{\hat{I}^+}(x^k + \Delta x^k)$. Finally, let μ be some small number, e.g. $\mu := 10^{-8}/\max\{1, \delta\}$. If

$$\|f_E^+\|_1 + \|f_{\hat{I}^+}^+\|_1 > 0.2\left(\|f_E\|_1 + \|f_{\hat{I}}\|_1\right) + 0.8t_k \tag{5}$$

the "second order correction problem"

$$\begin{array}{lrcrcl} \text{minimize} & & & \frac{\mu}{2}s^T s + t & & \\ & Df_E s & - & tf_E^+ & = & -f_E^+, \\ & Df_{\hat{I}+} s & - & tf_{\hat{I}+} & \leq & -f_{\hat{I}^+}, \\ \text{s.t.} & \hat{A}^+ s & & & \leq & \hat{b}^+, \\ & & & -t & \leq & 0, \\ & \|s\| & & & \leq & \delta \end{array} \tag{6}$$

is solved. The solution to this problem is denoted by s^k. If (5) does not hold (i.e. if the reduction of the constraint violation amounts to at least 80% of the predicted reduction of the step $x^k \mapsto x^k + \Delta x^k$) then set $s^k := 0$.

3.3.1 Choice of δ

Initially, δ_0 is set to some moderate value such as $\delta_0 = 1$. At each iteration, steps Δx^k, s^k are generated by (4), (6) with some trust region radius δ_k. The solutions to (4), (6) define a curve

$$x(\alpha) := x^k + \alpha \Delta x^k + \alpha^2 s^k$$

for $\alpha \in (0, 1]$. The next iterate $x^{k+1} := x(\alpha_k)$ is determined by a line search with respect to α. The choice of δ_{k+1} for the next iteration is intended as large as possible while anticipating steps of length $\alpha = 1$.

More precisely, an exact penalty function $\Theta : \mathbb{R}^n \to \mathbb{R}$ with

$$\Theta(x) := f(x) + \bar{M} \sum_{j \in E} |f_j(x)| + \bar{M} \sum_{i \in I} (f_i(x))_+$$

is used where $\bar{M}$ is some number larger than the estimates of the absolute values of the Lagrange multipliers. Observe that $Ax \leq b$, $A(x + \Delta x) \leq b$, and $A(x + \Delta x + s) \leq b$ also imply $Ax(\alpha) \leq b$ for $\alpha \in [0, 1]$ so that the feasibility with respect to linear inequalities remains satisfied.

The trust region radius δ_{k+1} for the next iteration is updated as follows:

- If $\alpha \in (0, 0.9)$ let $\delta_{k+1} = \alpha\|\Delta x\|_2$.
- If $\alpha \geq 0.9$ let $\delta_{k+1} = \max\{\delta_k,\ 2\alpha\|\Delta x\|_2\}$.
- (For numerical stability of the SOCP solver we also set $\delta_{k+1} = \min\{\delta_k,\ 10\|\Delta x\|_2\}$.)

3.4 Update of H_k

After each iteration the Hessian approximation of the Lagrangian is updated. This update is applied to the current approximation of the Hessian and not to its projection onto a subspace defined by the active constraints. In particular, if the multipliers are fixed and the set of active constraints is changing, this change will not influence the Hessian approximation.

Let $x := x^k$ and let x^+ be the short hand notation for the vector x^{k+1}. Set $\Delta x := x^+ - x$. (This notation coincides with the one of Section 3.3.1 if $\alpha_k = 1$ and $s^k = 0$; here it is used also for all other cases.) Likewise denote the finite difference approximations to $Df(x)$, $Df_E(x)$, and $Df_I(x)$ at x^k by Df, Df_E, and Df_I, and at x^+ by Df^+, Df_E^+, and Df_I^+. Denote the optimal multipliers of Problem (4) by y_E, y_I, and y_J. Then, set

$$g := Df^T + Df_E^T y_E + Df_I^T y_I \quad \text{and} \quad g^+ := (Df^+)^T + (Df_E^+)^T y_E + (Df_I^+)^T y_I.$$

These are the gradients of the Lagrangian at x^k and at x^+ except from the linear parts associated with y_J. The linear parts are omitted since they are constant and therefore cancel during the computation of Δg below. The estimate $H = H_k$ is updated to an estimate H^+ as follows:

For $\|\Delta x\| \leq dt$ set $H^+ = H$, but if $\|\Delta x\| > dt$ where dt is as in Section 3.1.1, then, the difference of the gradients g and g^+ at both points is used for a Quasi-Newton update based on the PSB formula. Let $\Delta g := g^+ - g$, $\Delta x = x^+ - x$, and

$$\Delta H := \frac{(\Delta g - H\Delta x)\Delta x^T + \Delta x(\Delta g - H\Delta x)^T}{\Delta x^T \Delta x} - \frac{(\Delta g - H\Delta x)^T \Delta x}{(\Delta x^T \Delta x)^2} \Delta x \Delta x^T.$$

The PSB update [12] is then given by setting

$$H := H + \Delta H.$$

3.5 Overall algorithm

The algorithm of the preceding subsections is summarized below:

Input: The data to Problem (1) and an initial point not necessarily satisfying any of the constraints.

Preprocessing: Convert (1) to a problem of the form (2) with a starting point x^0 satisfying the linear inequality constraints.

Initialization: Set $\delta_0 := 1$, $H_0 := 0$, and $k = 0$. Evaluate $f(x^0)$, $f_E(x^0)$, $Df_E(x^0)$, $f_I(x^0)$, and $Df_I(x^0)$.

Repeat

1. Project the Hessian H_k of the Lagrangian onto the null space of the gradients of the active constraints and generate a positive definite approximation $\tilde{H}_k$ to the projected Hessian.
2. Solve Problem (4).
3. Evaluate $f_E(x^k+\Delta x^k)$, $Df_E(x^k+\Delta x^k)$, $f_I(x^k+\Delta x^k)$, and $Df_I(x^k+\Delta x^k)$.
4. If (5) is satisfied solve (6), else set $s^k = 0$.
5. Determine α_k and update $x^{k+1} := x(\alpha_k)$ and δ_{k+1} as in Section 3.3.1.
6. Evaluate $Df_E(x^{k+1})$ and $Df_I(x^{k+1})$, update H_k as in Section 3.4, and set $k := k + 1$.

until some convergence criterion holds or until some iteration limit is reached.

In the numerical implementation several minor modifications of the above general concepts were made: Whenever the objective function was undefined, its value was reset to $+\infty$ to avoid early termination.

Constraints such as $x_i \leq 10^{20}$ imply very large entries in the right hand side of the SOCP subproblem and a low accuracy in the remaining components of the solution generated by the SOCP solver used in this paper. To prevent this effect, all constraints that are necessarily inactive were eliminated before the solution of each SQP subproblem by using bounds derived from the current trust region radius.

4 Numerical results

To provide some intuition about the convergence of the algorithm, some preliminary numerical examples on the CUTEst test set are reported in this section. All problems were selected for which $n + mi + mA \leq 500$ is satisfied where n is the number of variables, mi the number of nonlinear inequality constraints, and mA the number of linear inequality constraints. (The number of equality constraints was not considered for this selection.) The total running time including loading and processing times for all 651 test problems selected in this way was 4 hours 43 minutes on a recent desktop computer (Intel(R) Core(TM) i7-4770 CPU at 3.40GHz with four CPUs).

The dimensions "n" of these 651 problems have a mean value of 34 and a quadratic mean value of 86; among the 651 test problems, there are 3 problems of dimension 1, and 113 problems of dimension 2, and 13 problems of dimension ≥ 400.

We are not aware of any other public domain package for nonlinearly constrained optimization that does not require derivative informations and we do not attempt a comparison with highly sophisticated commercial solvers such as DNOPT (in finite difference mode) [5] or the Optimization toolbox of Matlab. The code of Sampaio and Toint [13] targets problems with expensive function evaluations and is not yet publicly distributed. We do not include it either in any comparison.

Table 1 reports the termination messages to the 651 selected problems. The maximum number of iterations was set $50n$ for these problems where n is the

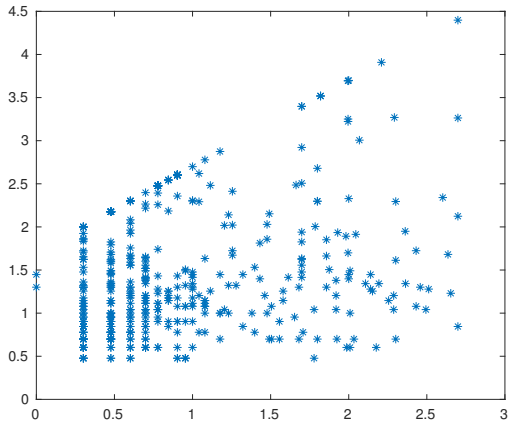

Fig. 4.1: Dimensions vs. number of iterations ($\log_{10}$-$\log_{10}$ -scale).

number of variables. For 117 problems the termination messages 4,5, or 6 indicate that no approximate first order optimality condition could be satisfied; for some of these 117 problems, however, progress over the initial point or near optimality was achieved without obtaining an approximate certificate. On the other hand, some problems in the CUTEst test set are deliberately given in an ill-posed form, e.g. more nonlinear equality constraints than variables. For such problems the starting point either satisfies the Fritz-John conditions or some infeasible stationarity conditions. Such situation will generally generate a termination message 1 or 2.

term. message	1	2	3	4	5	6
number of cases	287	167	80	58	58	1

Table 1, results for 651 CUTEst problems of dimension ≤ 500.

Termination message 6 occured for problem HS72 where the algorithm carried out 57 iterations initially increasing the constraint violation and not being able to reduce the constraint violation again.

There were 79 problems where the algorithm stopped after reaching the preset limit of 50n iterations, 3 of which with termination message 1, and 18 of which with termination message 3, and 58 of which with termination message 4.

There were 460 problems for which the algorithm took more than 2 iterations. Figure 4.1 displays the dimension n vs. number of iterations for these 460 problems on a $\log_{10}$-$\log_{10}$ scale. Apart from the upper bound of 50n iterations, a clear trend cannot be identified from this plot.

Note: For problem "RECIPE" the initial point does not satisfy the linear equality constraint. When projecting the initial point to eliminate the linear equality constraint, it turns out that one of the nonlinear constraint functions is not finite at the projected point, and the algorithm stops.

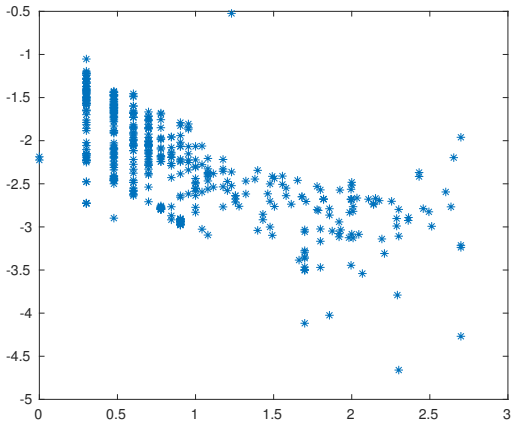

Fig. 4.2: Dimensions vs. $\frac{1}{n}\times$(time per iteration) – ($\log_{10}$-$\log_{10}$ -scale).

If such projection onto a point with undefined function values must be avoided, the linear equality constraints can be included within the nonlinear equality constraints. These will only be changed gradually during the algorithm – controlled by a line search for a merit function that does not generate iterates with undefined function values.

Figure 4.2 displays the dimension n vs. $\frac{1}{n}\times$(time per iteration) for these 460 problems, again on a log-log scale. The scaling $\frac{1}{n}$ reflects the fact that $2n$ function evaluations are performed at each iteration. Averaging these numbers, the time per iteration is roughly given by 0.025 n, slightly less for the larger problems and slightly more for the smaller ones (possibly due to the overhead of each function call and only when considering this particular test set!).

Acknowledgment

The first author would like to thank Helene Hibon for corrections of an earlier version of the code [7].

References

[1] R. H. Byrd: An example of irregular convergence in some constrained optimization methods that use the projected Hessian, Math. Programming 32, 232–237 (1985).

[2] R.H. Byrd, J. Nocedal: An analysis of reduced Hessian methods for constrained optimization, Math. Programming 49, 285–323 (1991).

[3] Celis, M., Dennis, J., Tapia, R.: "A trust region strategy for nonlinear equality constrained optimization", *Numerical Optimization* (P. Boggs, R. Byrd, and R. Schnabel, eds), SIAM, 71–82 (1985).

[4] The CUTEr/st Test Problem Set,
http://www.cuter.rl.ac.uk/Problems/mastsif.html

[5] Gill, P.E., Saunders, M.A., and Elizabeth Wong, E.: User's Guide for DNOPT: Software for Medium-Scale Nonlinear Programming, Center for Computational Mathematics, University of California, San Diego, La Jolla, CA, Center for Computational Mathematics Report (2016).

[6] C.B. Gurwitz, M.L. Overton: SQP methods based on approximating a projected Hessian matrix, SIAM J. Sci. Stat. Comp. 10, 631–653 (1989).

[7] F. Jarre, F. Lieder: Local Minimization Without Evaluating Derivatives, a Matlab collection. http://www.opt.uni-duesseldorf.de/en/forschung-fs.html (2017).

[8] F. Jarre and F. Lieder: The solution of Euclidean norm trust region SQP subproblems via second order cone programs: an overview and elementary introduction, Optimization Methods and Software (2017).

[9] M. Lazar, F. Jarre: Calibration by Optimization Without Using Derivatives, Optimization and Engineering, **17(4)**, 833–860 (2016).

[10] N. Maratos: Exact Penalty Function Algorithms for Finite Dimensional and Control Optimization Problems, Ph.D. Thesis, Imperial College of Science and Technology, London, UK (1978).

[11] J. Nocedal, M.L. Overton: Projected Hessian updating algorithms for nonlinearly constrained optimization, SIAM J. Numer. Anal. 22, 821–850 (1985).

[12] M.D.J. Powell: A new algorithm for unconstrained optimization, in Nonlinear Programming, J.B. Rosen, O.L. Mangasarian, and K. Ritter, eds., Academic Press, New York, 31-65 (1970).

[13] Ph. R. Sampaio and Ph. L. Toint: Numerical experience with a derivative-free trust-funnel method for nonlinear optimization problems with general nonlinear constraints, Optimization Methods and Software **31 (3)** 511–534 (2016).