Validation of an Augmented Lagrangian Algorithm with a Gauss-Newton Hessian Approximation Using a Set of Hard-Spheres Problems

Nataša Krejić [*] José Mario Martínez [†] Margarida Mello [†]

Elvio A. Pilotta [†]

January 9, 2001

Abstract

An Augmented Lagrangian algorithm that uses Gauss-Newton approximations of the Hessian at each inner iteration is introduced and tested using a family of Hard-Spheres problems. The Gauss-Newton model convexifies the quadratic approximations of the Augmented Lagrangian function thus increasing the efficiency of the iterative quadratic solver. The resulting method is considerably more ef-

[*]Institute of Mathematics, University of Novi Sad, Trg Dositeja Obradovića 4, 21000 Novi Sad, Yugoslavia, E-mail: natasa@unsim.im.ns.ac.yu. This author was supported by FAPESP (Grant 96/8163-9).

[†]Department of Applied Mathematics, IMECC-UNICAMP, University of Campinas, CP 6065, 13081-970 Campinas SP, Brazil. These authors were supported by PRONEX, FAPESP (Grant 90-3724-6), CNPq and FAEP-UNICAMP. E-mail: martinez@ime.unicamp.br, margarid@ime.unicamp.br, pilotta@ime.unicamp.br

ficient that the corresponding algorithm that uses true Hessians. A comparative study using the well-known package LANCELOT is presented.

1 Introduction

In recent years we have been involved with the development of algorithms based on sequential quadratic programming [11] and inexact restoration [16, 18] for minimization problems with nonlinear equality constraints and bounded variables.

The validation of these algorithms require their comparison with well established computer methods for the same type of problems, which include methods of the same family (as other SQP methods in the first case and GRG like methods in the second) as well as methods that adopt a completely different point of view, as is the case of Penalty and Augmented Lagrangian algorithms. The most consolidated practical Augmented Lagrangian method currently available is the one implemented in the package LANCELOT, described in [4]. This was the method used, for example, in [11], to test the reliability of a new large-scale sequential quadratic programming algorithm.

In the course of the above mentioned experimental studies we felt the necessity of intervening in the Augmented Lagrangian code in a more active way than the one permitted to users of LANCELOT. As a result of this practical necessity, we became involved with the development of a different Augmented Lagrangian code, which preserves most of the principles of the LANCELOT philosophy, but also has some important differences.

Following the lines of [4], a modern Augmented Lagrangian method is essentially composed by three nested algorithms:

- The external algorithm updates the Lagrange multipliers and the penalty parameters, decides stopping criteria for the internal algorithm and the rules for declaring convergence or failure of the overall procedure.

- An internal algorithm minimizes the augmented Lagrangian function with bounds on the variables. Trust region methods, where the subproblem consists on the minimization of a quadratic model on the intersection of two boxes, the one that defines the problem and the trust-region box, are used both in [4] and in our implementation.

- A third algorithm deals with the resolution of the quadratic subproblem. While LANCELOT restricts its search to the face determined by an approximate Cauchy point, our code explores the domain of the subproblem as a whole.

The second item, specifically where it deals with the formulation of the quadratic subproblem, is the one in which we felt more strongly the desire to intervene. On one hand, we tried many alternative sparse quasi-Newton schemes (without success, up to now). On the other hand, we used a surprisingly effective simplification of the true Hessian of the Lagrangian, called, in this paper, "the Gauss-Newton Hessian approximation" by analogy with the Gauss-Newton method for nonlinear least-squares, which can be interpreted as the result of excluding from the Hessian of a sum of squares those terms involving Hessian of individual components.

In order to validate our augmented Lagrangian implementation we selected a family of problems in which we have particular interest, known as the family of Hard-Spheres problems.

The Hard-Spheres Problem belongs to a family of sphere packing problems, a class of challenging problems dating from the beginning of the seventeenth century.

In the tradition of famous problems in mathematics, the statements of these problems are elusively simple, and have withstood the attacks of many worthy mathematicians (e.g. Newton, Hilbert, Gregory), while most of its instances remain open problems. Furthermore, it is related to practical problems in chemistry, biology and physics, see, for instance, the list of examples in [19], concerning mainly three-dimensional problems, or peruse the 1550-item-long bibliography in [5]. The *Hard-Spheres Problem* is to maximize the minimum pairwise distance between p points on a sphere in $\mathbb{R}^n$. This problem may be reduced to a nonlinear optimization problem that turns out, as might be expected from the mentioned history, to be a particularly hard, nonconvex problem, with a potentially large number of (nonoptimal) points satisfying KKT conditions. We have thus a class of problems indexed by the parameters n and p, that provides a suitable set of test problems for evaluating Nonlinear Programming codes.

Very convenient is the fact that the Hard-Spheres Problem may be regarded as the feasibility problem associated with another famous problem in the area, the *Kissing Number Problem*, which seeks to determine the maximum number $\mathcal{K}_n$ of nonoverlapping spheres of given radius in $\mathbb{R}^n$ that can simultaneously touch (kiss) a central sphere of same radius. Thus, if the distance obtained in the solution of the Hard-Spheres Problem, for given n and p, is greater than or equal to the radius of the sphere on which the points lie, one may conclude that $\mathcal{K}_n \geq p$. We use the known solution of the three-dimensional Kissing Number Problem to calibrate our code, described below, and choose for testing the code values of n, p that might bring forth new knowledge about the problem, or strengthen existing conjectures about the true (but, alas, not rigorously established) value of $\mathcal{K}_n$, from the following table of known values/bounds of $\mathcal{K}_n$ given in [5]:

n	$\mathcal{K}_n$
1	2
2	6
3	12
4	24–25
5	40–46
6	72–82
7	126–140
8	240
9	306–380
10	500–595
11	582–915
12	840–1416

Table 1: Known values/bounds of $\mathcal{K}_n$.

This paper is organized as follows. In Section 2 we formulate the Hard-Spheres Problem as a nonlinear programming problem and we relate the main characteristics of ALBOX, our Augmented Lagrangian Algorithm. In Section 3 we explain how the main algorithmic parameters of ALBOX were chosen. (Here we follow a previous study in [15].) In Section 4 we introduce the Gauss-Newton Hessian approximation and discuss the effect of its use in comparison with the use of true Hessians of the Lagrangians. In Section 5 we describe the parameters used with LANCELOT. The numerical experiments, obtained by running ALBOX and LANCELOT for a large number of Hard-Spheres problems, are presented in Section 6. Finally, some conclusions are drawn in Section 7.

2 ALBOX

The straightforward formulation of the Hard-Spheres Problem leads to the following maxmin problem, where r is the radius of the sphere, centered on the origin, on which the points lie:

$$\begin{array}{ll} \max \ \ \min_{i \neq j} & \|y^i - y^j\| \\ \text{s.t.} & \|y^k\| = 2r, \quad k = 1, \ldots, p. \end{array} \tag{1}$$

The vectors y^k belong to $\mathbb{R}^n$ and $\|\cdot\|$ is the Euclidean norm. Since the answer to the problem is invariant under the choice of positive r, we let $r = 1/2$. Furthermore, using the definition of $\langle \cdot, \cdot \rangle$, the standard inner product in $\mathbb{R}^n$, and the constraints, it is easy to see that (1) is equivalent to

$$\begin{array}{ll} \min \ \ \max_{i \neq j} & \langle y^i, y^j \rangle \\ \text{s.t.} & \|y^k\| = 2r, \quad k = 1, \ldots, p. \end{array} \tag{2}$$

Applying the classical trick for transforming minimax problems into constrained minimization problems, we reduce (2) to the nonlinear program

$$\begin{array}{lrcl} \min & z & & \\ \text{s.t.} & z & \geq & \langle y^i, y^j \rangle, \quad \forall\ i \neq j, \\ & \|y^k\| & = & 1, \quad k = 1, \ldots, p. \end{array} \tag{3}$$

Adding slack variables to the first set of constraints and squaring the second set of equations in order to avoid nonsmoothness in the first derivatives, we obtain

$$\begin{array}{lccccccl} \min & z & & & & & & \\ \text{s.t.} & z & - & \langle y^i, y^j \rangle & - & w_{ij} & = & 0, \quad \forall\ i \neq j, \\ & & & \|y^k\|^2 & & & = & 1, \quad k = 1, \ldots, p, \\ & & & & & w & \geq & 0. \end{array} \tag{4}$$

which is of the general form

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & h(x) = 0 \\ & \ell \leq x \leq u. \end{aligned} \tag{5}$$

ALBOX, the augmented Lagrangian code developed, approximately solves

$$\begin{aligned} \min \quad & L(x, \lambda, \rho) \\ \text{s.t.} \quad & \ell \leq x \leq u, \end{aligned} \tag{6}$$

at each Outer Iteration, where

$$L(x, \lambda, \rho) = f(x) + \sum_i \lambda_i h_i(x) + \sum_i \rho_i h_i^2(x) \tag{7}$$

is the augmented Lagrangian function associated with (5), λ is the current approximation to the Lagrange multipliers and ρ (≥ 0) is the current vector of penalty parameters. These are updated at the end of the Outer Iteration.

Subproblem (6) is solved using BOX, the box-constrained solver described in [10]. This iterative method minimizes a quadratic approximation to the objective function on the intersection of the original feasible set, the box $\ell \leq x \leq u$, and the trust region (also a box), at each iteration. If the original objective function is sufficiently reduced at the approximate minimizer of the quadratic, the corresponding trial point is accepted as the new iterate. Otherwise, the trust region is reduced. The main algorithmic difference between BOX and the method used in [2] is that in BOX the quadratic is explored on the whole intersection of the original box and the trust region whereas in [2] only the face determined by an "approximate Cauchy point" is examined.

ALBOX is a Double Precision FORTRAN 77 code that aims to cope with large-scale problems. For this reason, factorization of matrices is not used at all. The quadratic solver used to solve the subproblems of the box-constraint algorithm, QUACAN, visits the different faces of its domain using conjugate gradients on the interior of each face and “chopped gradients” as search directions to leave the faces. We refer the reader to [1], [9] and [10], for details on the actual implementation of QUACAN. In most iterations of this quadratic solver, a matrix-vector product of the Hessian approximation and a vector is computed. Occasionally, an additional matrix-vector product may be neccessary.

The performance of ALBOX, and, in fact, of most sophisticated algorithms, depends on the choice of many parameters. The most sensitive parameters were adjusted using the Kissing Problem with $n = 3$ and $p = 12$ (*Icosahedron Problem*). We discuss these choices in the next section. A similar analysis was carried out for LANCELOT, and is described in section 5.

3 Choice of parameters for ALBOX

3.1 Penalty parameters and Lagrange multipliers

The vector ρ of penalty parameters associated with the equality constraints $h(x) = 0$ are updated after each Outer Iteration. We considered two possiblities: to update each component according to the decrease of the corresponding component of $h(x)$ or using a global criterion based on $h(x)$. The specific alternatives contemplated were, assuming x to be the initial point at some Outer Iteration and $\bar{x}$ the final one:

1. increase ρ_i only if $|h(\bar{x})_i|$ is not sufficiently smaller than $|h(x)_i|$;

2. increase ρ_i only if $\|h(\bar{x})\|_\infty$ is not sufficiently smaller than $\|h(x)\|_\infty$.

Preliminary experiments revealed, perhaps surprisingly, that the "global strategy" 2 is better than the first. In fact, when ρ_i is not updated, but the other components of ρ are, the feasibility level $|h(\bar{x})_i|$ tends to deteriorate at the next iteration and, consequentely, a large number of Outer Iterations becomes necessary. In other words, it seems that a strategy based on 1 encourages a zigzagging behavior, with successive iterates alternatingly satisfying one constraint or another. Thus, although the original formulation allows for one penalty parameter for each equality constraint, in practice it is as if we worked with one parameter for all of them, since they are all initialized at the same value (tests indicate that 10 is an adequate initial value) and are all updated according to the same rule (once again based on tests, they are increased by a factor of 10 when sufficient improvement of feasibility is not detected). Here we considered that "a sufficiently smaller than b" means that $a \leq 0.01b$.

It must be pointed out that the behavior of penalty parameters is not independent of the strategy for updating the Lagrange multipliers. With algorithmic simplicity in mind, we adopted a "first order formula". Letting $\bar{\lambda}$ be the Lagrange multiplier at the start of a new Outer Iteration and λ, ρ be the Lagrange multipliers and penalty parameters at the previous iteration, we set

$$\bar{\lambda}_i = \lambda_i + \rho_i h(\bar{x})_i$$

for all $i = 1, \ldots, m$. Initially, $\lambda = 0$.

3.2 Stopping criteria for box-constraint solver

Each Outer Iteration ends when one of the several stopping criteria for the algorithm that solves the augmented Lagrangian box-constrained minimization problem (6) is reached. There is the usual maximum number of iterations safeguard, which is set at 100 for QUACAN calls.

Other than that, we consider that the box-constraint algorithm BOX converges when

$$\|g_P(x)\|_2 \leq \varepsilon,$$

where $g_P(x)$ is the "continuous projected gradient" of the objective function of (6) at the point x. This vector is defined as the difference between the projection of $x - \nabla L(x, \lambda, \rho)$ on the box and the point x. The tolerance ε may change at each Outer Iteration. We tested two strategies for ε: one that defines ε dynamically depending on the degree of feasibility of the current iterate and another that fixes ε at 10^{-5}. Althought not conclusive, results for the Icosahedron Problem were better when the constant ε strategy was used. This was, therefore, the strategy adopted for further tests. Incidentally, the opposite was adopted in [8], where a similar Augmented Lagrangian Algorithm was used to solve linearly constrained problems derived from physical applications. Theoretical justifications for the inexact minimization of subproblems in the augmented Lagrangian context can also be found in [12, 13].

The box-constraint code admits other stopping criteria. For instance, execution may stop if the progress during some number of consecutive iterations is not good enough or if the the radius of the trust region becomes too small. Nevertheless, best results were obtained inhibiting these alternative stopping criteria.

3.3 Parameters for the quadratic solver

QUACAN is the code called to minimize quadratic functions (augmented Lagrangians in this case) subject to box constraints. Its efficiency, or lack thereof, plays a crucial role in the overall behavior of the Augmented Lagrangian Algorithm. Its parameters must therefore be carefully chosen.

Firstly we examine the convergence criterion. If the projected gradient of the quadratic is null, the corresponding point is stationary. Accordingly, convergence is considered achieved when the norm of this projected gradient is less than a fraction of the corresponding norm at the initial point. In this case, we use "non-continuous projected gradients," in which the projections are not computed on the feasible box but on the active constraints. Fractions 1/10, 1/100 and 1/100000 were tested on the Icosahedron Problem, and the first choice provided the best behavior, being the one employed subsequently.

The maximum number of iterations allowed is also an important parameter, since otherwise we may invest too much effort solving problems only distantly related to the original one. We found that the number of variables of the problem, $np+\binom{p}{2}+1$, is a suitable delimiter in this case. Other non-convergence stopping criteria were inhibited.

The radius of the trust region determines the size of the auxiliary box used in QUACAN. The nonlinear programming algorithm is sensitive to the choice of δ, the first trust region radius. After testing different values, we selected $\delta = 10$ as an appropriate choice.

Another important parameter is $\eta \in (0,1)$, the parameter that determines whether the next iterate must belong to the same face as the current one, or not. Roughly speaking, if η is small, the algorithm tends to leave the current face as soon as a mild decrease of the quadratic is detected. On the other hand, if $\eta \approx 1$, the algorithm only abandons the current face when the current point is close to a stationary point of the quadratic on that face. A rather surprising result was that, for the Icosahedron Problem, the conservative value $\eta = .95$ was better than smaller values.

Finally, when the quadratic solver hits the boundary of its feasible region, an extrapolation step may be tried, depending on the value of the extrapolation parameter $\gamma \geq 1$. If γ is large, new points will be tried at which the number of active bounds may be considerably increased. No extrapolation is tried when $\gamma = 1$. Tests indicated that $\gamma = 10$ is a convenient choice for the Hard-Spheres Problem.

4 Approximate Hessian

The nonlinear optimization problem (4) obtained in section 2 is the version of the Hard-Spheres Problem that was chosen for our tests. It was pointed out that (4) is of the general form

$$\begin{array}{lrcl} \min & f(x) & & \\ \text{s.t.} & h(x) & = & 0 \\ & \multicolumn{3}{l}{\ell \leq x \leq u.} \end{array}$$

whose associated augmented Lagrangian is

$$L(x, \lambda, \rho) = f(x) + \langle \lambda, h(x) \rangle + \frac{\rho}{2} \|h(x)\|_2^2.$$

Thus

$$\nabla L(x, \lambda, \rho) = \nabla f(x) + \sum_{i=1}^{m} \lambda_i \nabla h_i(x) + \rho h'(x)^T h(x)$$

and

$$\nabla^2 L(x, \lambda, \rho) = \nabla^2 f(x) + +\rho h'(x)^T h'(x) + \sum_{i=1}^{m} [\lambda_i + \rho h_i(x)] \nabla^2 h_i(x).$$

Although $\nabla^2 L(x, \lambda, \rho)$ tends to be positive definite when ρ is large, λ is close to the correct Lagrange multipliers and x is close to a solution, this is not the case at the early stages of augmented Lagrangian calculations. On the other hand, the simplified matrix obtained by neglecting the term involving second order derivatives of the constraint functions

$$B(x, \rho) = \nabla^2 f(x) + +\rho h'(x)^T h'(x)$$

is always positive semidefinite in our case, independently of ρ and x. Of course, this is always the case when f is a convex function.

Another insight into $B(x, \rho)$ is provided by examining the problem

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & h'(z)(x - z) + h(z) = 0 \\ & \ell \leq x \leq u, \end{array} \tag{8}$$

where z is the current point being used in a BOX iteration. Problem (8) is obtained by replacing the original $h(x) = 0$ constraints with its first order (linear) approximation. But $B(z, \rho)$ happens to be the Hessian of the augmented Lagrangian associated with (8) at z! Furthermore, both the augmented Lagrangian associated with (8) and its gradient evaluated at z coincide with their counterparts associated with the original problem (4), evaluated at z.

The matrix vector products $\nabla^2 L(x, \lambda, \rho)v$ and $B(x, \rho)v$ seem cumbersome to compute at a first glance. But taking advantadge of their structure enables the computation to be done in $O(np)$ time.

In principle, using the true Hessian of the Lagrangian should the best possible choice, since it represents better the structure of the true problem. However, available algorithms for minimizing quadratics in convex sets are much more efficient when the quadratic is convex than otherwise. QUACAN is not an exception to this rule. Therefore, in the interest of improving the overall performance of the augmented Lagrangian algorithm, we decided to use $B(x, \rho)$ as Hessian Lagrangian approximation.

The results were indeed impressive. Table 2 lists the average statistics obtained for four of the eighteen test sets, where each (n, p) pair was run for fifty random starting points. The average number of Outer iterations, BOX iterations, Function evaluations, Matrix Vector Products, CPU time in seconds and minimum distance are given for the runs using the exact Hessian (first row of each set) and the ones using the approximate Hessian (second row). The minimum distances obtained were very close and on some instances the minimum distance obtained using the approximate Hessian was smaller than the one obtained using the exact Hessian. While the number of Outer iterations does not differ very much from one choice to the other, the number of BOX iterations and, consequently, the number of Matrix Vector Products sensibly decreases. The overall result is a marked decrease in CPU time. In Figure 1 we plot the average CPU times, for all eighteen tests, using the exact Hessian versus times using the approximate Hessian. Also shown is the line that gives the best fit of the data by a linear (not affine) function, namely $y = 0.374138\,x$, that is, the approximate Hessian option implies in a decrease of almost two thirds in CPU times.

5 Choice of parameters for LANCELOT

LANCELOT allows for the choice of exact or approximate first and second order derivatives. However, LANCELOT's manual [3] (p.111) "strongly recommends the use of exact second derivatives whenever they are available", and, on the other hand, there is no provision for an user supplied Hessian approximation. In fact we ran a

Problem size			Outer	Box	Funct.	MVP	CPU	Min
$\begin{bmatrix} n \\ p \end{bmatrix}$	var.	constr.	it.	it.	eval.		time	dist.
$\begin{bmatrix} 3 \\ 10 \end{bmatrix}$	76	55	4.86	37.06	52.14	1564.36	0.765	1.086487225412
			4.64	34.74	45.52	1194.70	0.476	1.083236334520
$\begin{bmatrix} 4 \\ 22 \end{bmatrix}$	320	253	4.60	91.10	123.90	16079.36	33.440	0.997314349536
			4.34	78.02	97.40	11222.14	20.032	0.997809583865
$\begin{bmatrix} 5 \\ 37 \end{bmatrix}$	852	703	5.00	274.10	358.54	142683.34	963.537	0.998632681285
			4.56	160.02	193.14	67020.22	373.141	0.998675348042

Table 2: Running ALBOX with exact (first row) and approximate Hessian (second row).

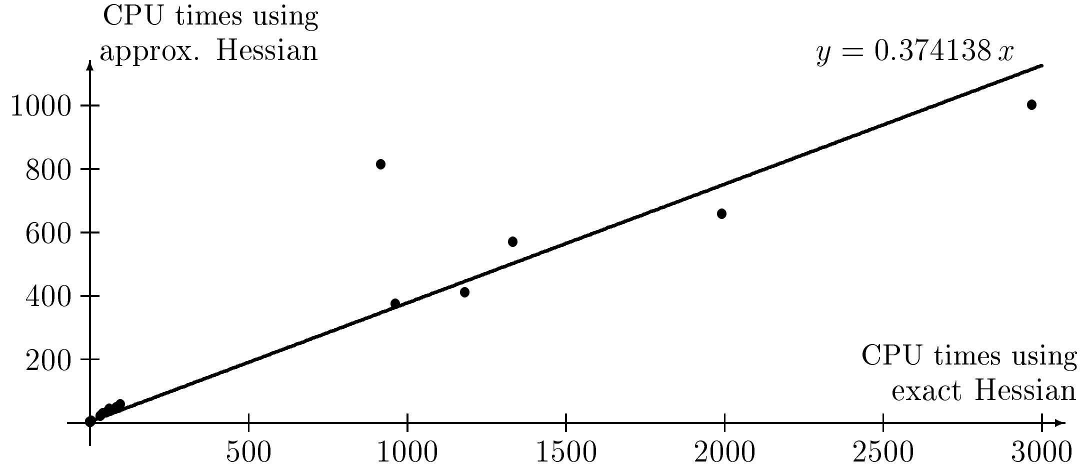

Figure 1: CPU times using exact Hessian (x-axis) versus using approximate Hessian (y-axis).

few tests with the default approximation (SR1) but the results were worse than those obtained using exact second derivatives, and thus this was the option adopted for all further tests. In the light of the experiments described in the previous section, this provides corroborating evidence to the effect that general purpose, consolidated packages, designed to provide a good performance with little interference from the user, may be more convenient to use than open ended, low-level interface codes, such as ALBOX; but, for the user willing to "get his hands dirty" the latter rawer code might not only prove competitive, it may actually outperform the former code, with its more polished though restrictive finish.

We also experimented with several different options for solving the linear equation solver, namely, without preconditioner, with diagonal preconditioner and with a band matrix preconditioner. The best results were obtained with the first option (no preconditioner). Another choice that slowed the algorithm, without noticeable improve the quality of solution, was requiring that the exact Cauchy point be computed. We settled to use the inexact Cauchy point option. The maximum number of iterations allowed is 1000. Finally, the gradient and constraints tolerances were the same chosen for ALBOX, namely 10^{-8}. The FORTRAN compiler option adopted for LANCELOT and ALBOX was "-O".

6 Numerical experiments

Tests were run on a Sun SparcStation 20, with the following main characteristics: 128Mbytes of RAM, 70MHz, 204.7 mips, 44.4 Mflops. Results for the fifty runs for each $(n,\ p)$ pair are summarized in the following tables. Table 3 summarizes the statistics that are "machine independent," typically involving number of iterations, number of function evaluations, with the exception of the optimal distances found. Quotes are needed because this is not completely accurate, since these numbers will in fact depend on machine precision, compiler manufacturer, and the like. Nevertheless, they certainly provide more independent grounds for comparison than CPU times, presented in Table 4, along with optimal distances.

Table 3 presents the mininum, maximum and average amounts of Outer and BOX iterations, function evaluations, Quacan iterations and matrix-vector-products/conjugate-gradient iterations (for Box and LANCELOT, respectively). First row of each set corresponds to ALBOX and second to LANCELOT. Unfortunately the only statistics available for both is the number of function evaluations. We paired the number of matrix-vector-products (MVP) output by ALBOX with the number of conjugate-gradient iterations (CGI) produced by LANCELOT, since each conjugate-gradient iteration involves a matrix-vector-product.

Although the algorithms behave very differently timewise, as we will shortly see, this is not a direct consequence of the number of function evaluations each performs. The best least-squares fit by a first degree polynomial gives $y = 5.74631 + 0.855356\,x$, where y is the number of function evaluations of ALBOX and x is the corresponding amount for LANCELOT, whereas a similar fit involving CPU times will give a coefficient of less than a third. On Figure 2 we plot the function evaluation pairs for all eighteen instances along with the best fit obtained.

Further still from providing an explanation for the higher efficiency of ALBOX is the comparison of MVP versus CGI. In this case the best fit gives $y = -1320.36 + 1.10655x$, where y is the number of MVP and x is the number of CGI. This suggests that, although both iterations involve a matrix-vector-product, a CGI is substantially costlier, timewise, than the MVP performed in ALBOX. A main

Table 3: Put first page of Table 3 here

Table 3: Put second page of Table 4 here

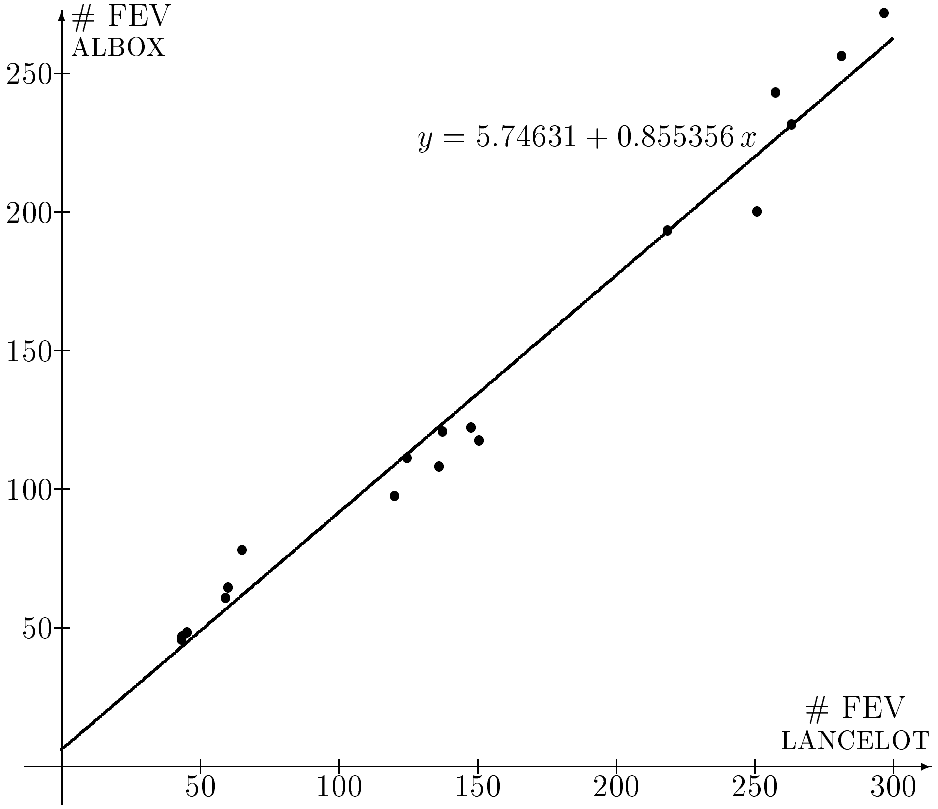

Figure 2: Number of function evaluations of LANCELOT versus ALBOX.

factor for this is that the matrix-vector-product in LANCELOT's conjugate gradient iteration deals with the true Hessian, whereas the one in ALBOX involves the approximate (and simpler) Hessian. Figure 3 contains the line corresponding to the best linear fit and the position of the (CGI, MVP) pairs.

Next we have Table 4, that presents similar statistics involving the optimal distances encountered and the CPU times, in seconds. The first (resp., second) row for each (n, p) pair gives the numbers obtained by ALBOX (resp., LANCELOT).

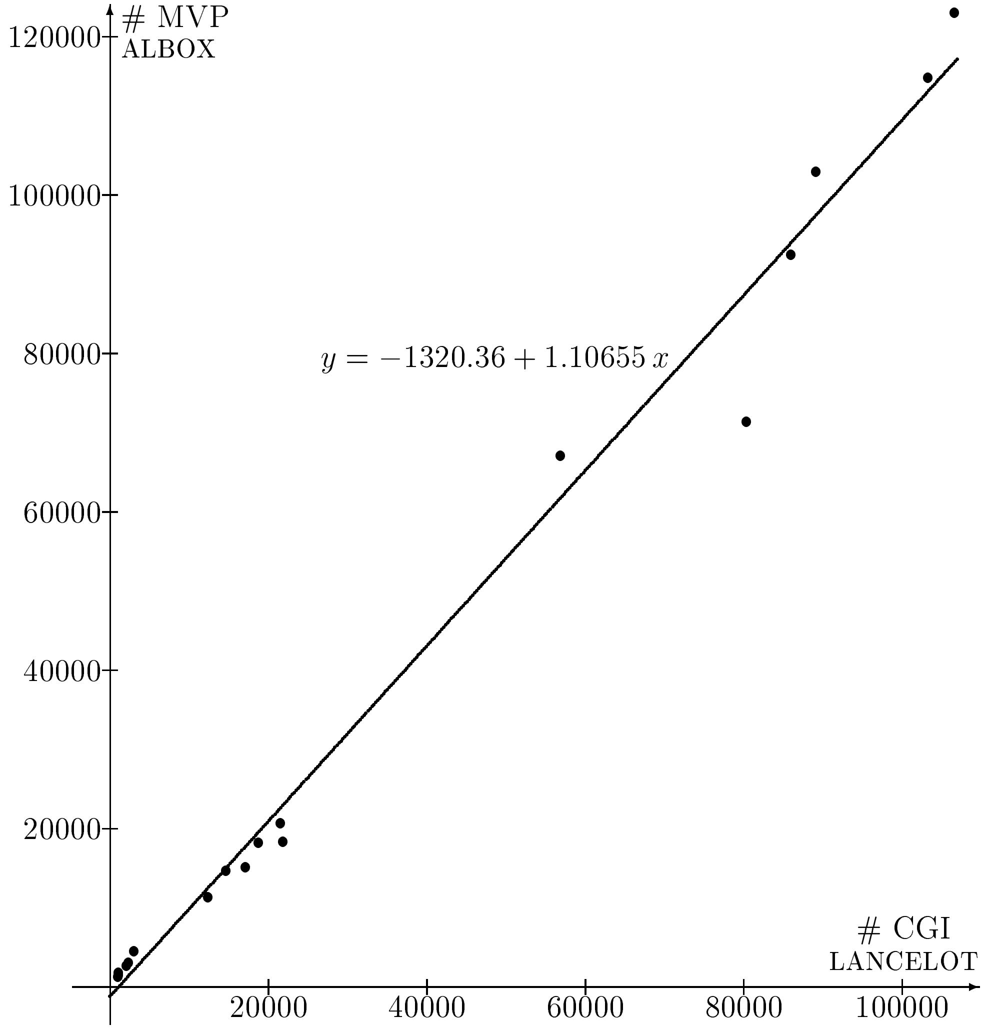

Figure 3: Number of CGIs of LANCELOT versus number of MVPs of ALBOX.

Table 4: Put Table 4 here

The information contained in Table 4 is depicted graphically below. The intervals (min., max) of distances/CPU times are represented by vertical segments, the averages are indicated with a diamond symbol for ALBOX and a bullet for LANCELOT. Graphs on the left refer to distances whereas graphs on the right refer to CPU times.

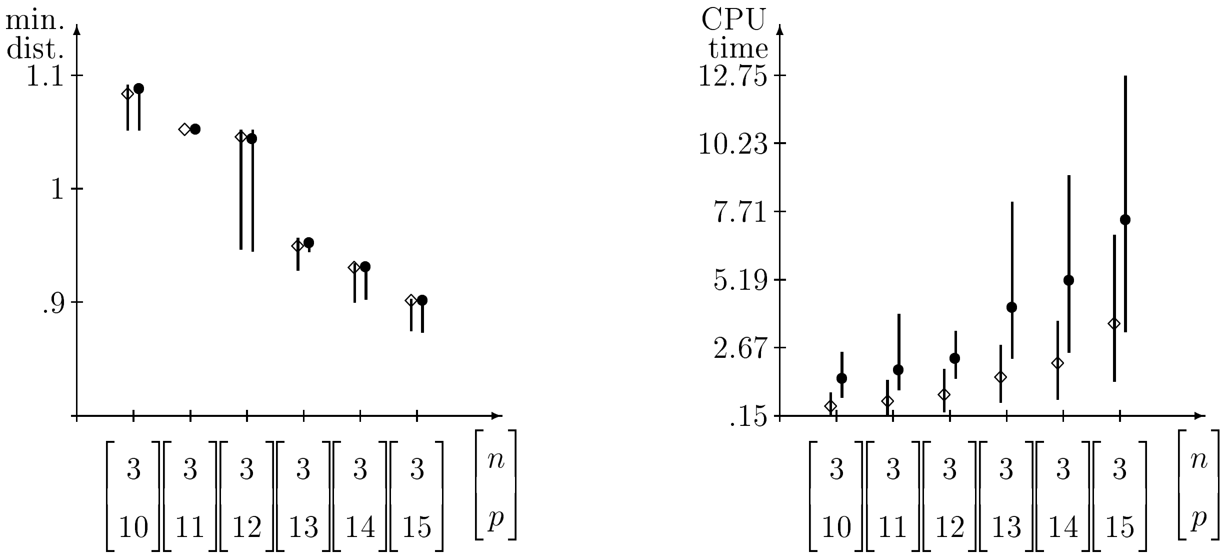

Figure 4: ALBOX (◇) and LANCELOT (●) results for $n = 3$.

The graphs in Figures 4–6 evidence the qualitative relative behavior of both codes. Notice that the diamonds and bullets are always close together in the graphs on the left, indicating that the quality of the optimal solutions obtained by both codes is similar. On the other hand, the bullets rise faster than the diamonds on the graphs on the right, which means that the CPU times for LANCELOT tend to be higher than those for ALBOX. The linear fit of ALBOX CPU times versus LANCELOT CPU times, $y = 0.31054\,x$—the coefficient is less than one third—, ploted in Figure 7 confirms this.

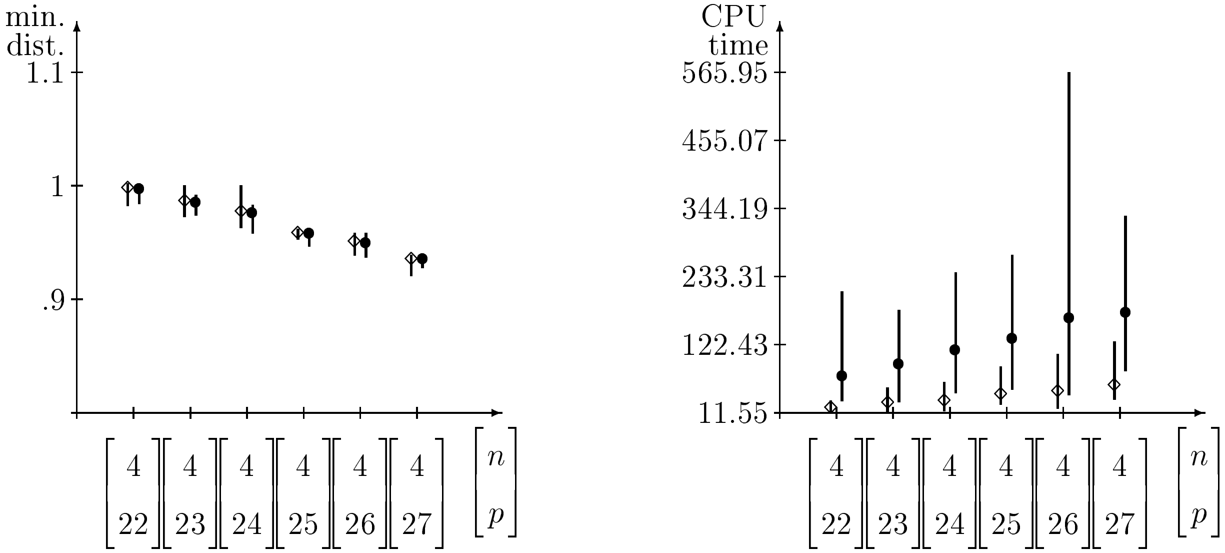

Figure 5: ALBOX (◇) and LANCELOT (●) results for $n = 4$.

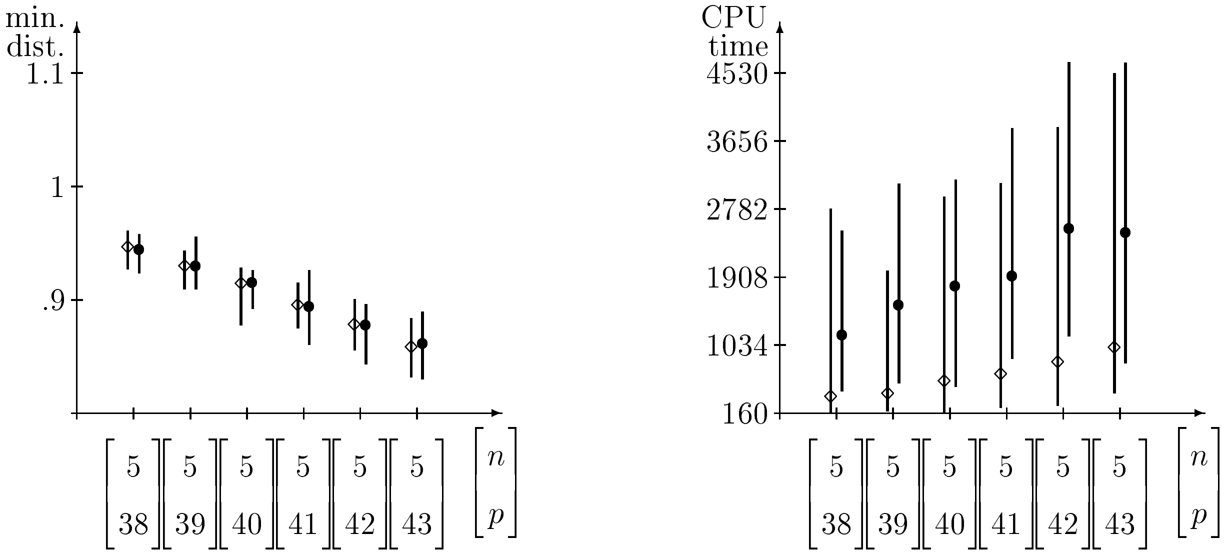

Figure 6: ALBOX (◇) and LANCELOT (●) results for $n = 5$.

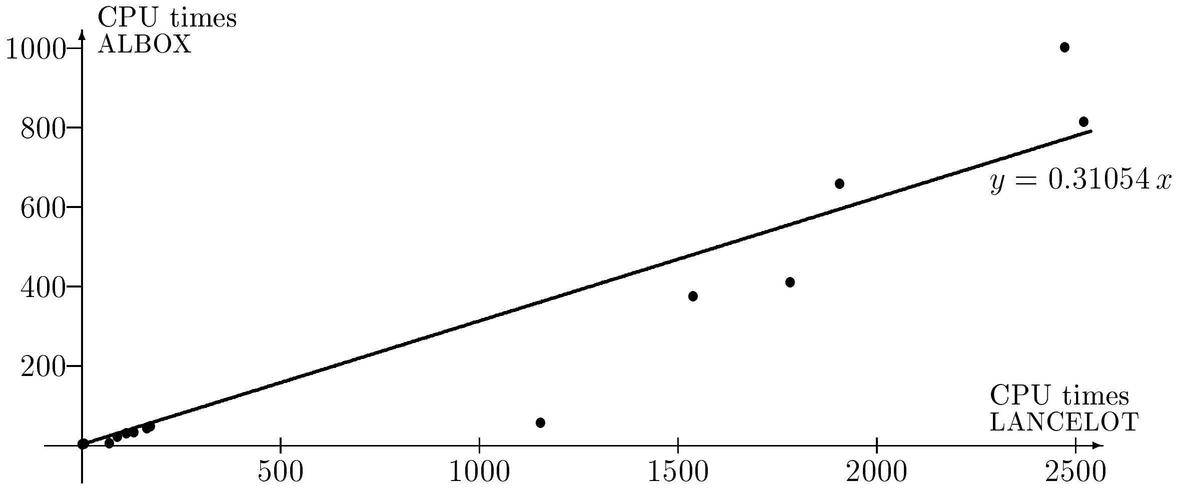

Figure 7: CPU times of LANCELOT versus those of ALBOX.

Finally, it should be noted that CPU times increase sharply as a function of problem size (represented, for instance, by the number of constraints). We tried several fits (linear, quadratic, exponential) and, though none seemed to provide a very good model for the data, the quadratic fit was the best one.

7 Conclusions

The main aspects of the Augmented Lagrangian methodology for solving large-scale nonlinear programming problems have been consolidated after the works of Conn, Gould and Toint which gave origin to the LANCELOT package. This algorithmic framework has been very useful in the last ten years for solving practical problems and for comparison purposes with innovative nonlinear programming methods. Very likely, this tendency will be maintained in the near future.

The present research was born as a result of our need to have more freedom in the formulation and resolution of the quadratic subproblems that arise in the LANCELOT-like approach to the Augmented Lagrangian philosophy. On one hand, we decided to exploit more deeply the whole trust region by means of the use of a box-constraint quadratic solver. On the other hand, perhaps more importantly, we tested a Gauss-Newton convex simplification of the quadratic model which turned out to be much more efficient than the straight Newton-like version of this model. Behind this gain of efficiency is the fact that the quadratic solver, though able to deal with nonconvex models, is far more efficient when the underlying quadratic has a positive semidefinite Hessian. It is usual, in Numerical Analysis, that a decision on the implementation of a high level algorithm depends on the current technology for solving low-level subproblems. It must only be warned that such a decision could change if new more efficient algorithms for solving the subproblems (nonconvex quadratic programming in our case) are found.

Our main objective is to use ALBOX, not only for solving real-life problems, but also for testing alternative nonlinear programming methods against it. We feel that having a deep knowledge of the implementation details of the code will enable us to be much more exacting when testing new codes, since it will be possible to fine tune the standard against which the new code is tested. The present study, apart from calling the reader's attention to convex simplified Gauss-Newton like subproblems, had the objective of validating our code, by means of its comparison with LANCELOT, using a set of problems that have an independent interest. The result of this comparison seems to indicate that ALBOX can be used as a competitive tool for nonlinear programming calculations.

References

[1] R.H. Bielschowsky, A. Friedlander, F.M. Gomes, J.M. Martínez and M. Raydan. An adaptative algorithm for bound constrained quadratic minimization. Technical report, Institute of Mathematics, State University of Campinas, 1995.

[2] A.R. Conn, N.I.M. Gould and Ph.L. Toint. A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Numerical Analysis*, 28:545–572, 1991.

[3] A.R. Conn, N.I.M. Gould and Ph.L. Toint. *LANCELOT, a Fortran Package for Large-Scale Nonlinear Optimization (Release A).* Springer-Verlag, Berlin, 1992.

[4] A. R. Conn, N. I. M. Gould and Ph. L. Toint. Global convergence of a class of trust region algorithms for optimization with simple bounds. *SIAM Journal on Numerical Analysis* 25:433–460, 1988. See also *SIAM Journal on Numerical Analysis* 26:764–767, 1989.

[5] J.H. Conway and N.J.C. Sloane. *Sphere Packings, Lattices and Groups.* Springer-Verlag, New York, 1988.

[6] K. Devlin. *Mathematics: The Science of Patterns.* Scientific American Library, New York, 1994.

[7] M.A. Diniz-Ehrhardt, M.A. Gomes-Ruggiero and S.A. Santos. Comparing the numerical performance of two trust-region algorithms for large-scale bound-constrained minimization. In R.J.B. Sampaio and J.Y. Yuan, editors, *Interna-*

tional Workshop of Numerical Algebra and Optimization, pages 23–24. Department of Mathematics, Universidade Federal do Paraná, Brazil, 1997.

[8] Z. Dostál, A. Friedlander and S.A. Santos. Augmented Lagrangians with adaptative precision control for quadratic programming with equality constraints. To appear in *Computational Optimization and Applications.*

[9] A. Friedlander and J.M. Martínez. On the maximization of a concave quadratic function with box constraints. *SIAM Journal on Optimization*, 4:177–192, 1994.

[10] A. Friedlander, J.M. Martínez and S.A. Santos. A new trust-region algorithm for bound constrained minimization. *Applied Mathematics and Optimization*, 30:235–266, 1994.

[11] F. M. Gomes, M. C. Maciel and J.M. Martínez. Nonlinear programming algorithms using trust regions and augmented Lagrangians with nonmonotone penalty parameters. To appear in *Mathematical Programming.*

[12] W. W. Hager. Analysis and implementation of a dual algorithm for constraint optimization. *Journal of Optimization Theory and Applications* 79:427–462, 1993.

[13] W. W. Hager. Dual techniques for constraint optimization. *Journal of Optimization Theory and Applications* 55:37–71, 1987.

[14] N. Maculan, P. Michelon and J. MacGregor Smith. Bounds on the kissing numbers in $\mathbb{R}^n$: mathematical programming formulations. Technical report, COPPE, University of Rio de Janeiro, Brazil, 1996.

[15] J.M. Martínez. Augmented Lagrangians and the resolution of packing problems. Technical Report 08/97, Institute of Mathematics, University of Campinas, Brazil.

[16] J.M. Martínez. A Two-Phase Model Algorithm with global convergence for Nonlinear Programming. to appear in *Journal of Optimization Theory and Applications*, Vol. 96, Feb. 1998.

[17] S.G. Nash. Preconditioning of truncated-newton methods. *SIAM Journal on Scientific and Statistical Computing*, 6:599–616, 1985.

[18] E. A. Pilotta. Linearly constrained spectral gradient methods and inexact restoration subproblems for nonlinear programming. Ph.D. Thesis, in preparation.

[19] E.B. Saff and A.B.J. Kuijlaars. Distributing many points on a sphere. *Mathematical Intelligencer*, 19:5–11, 1997.