Crash Start of Interior Point Methods

Jacek Gondzio*

School of Mathematics
The University of Edinburgh
Edinburgh EH9 3JZ, United Kingdom
Technical Report ERGO-2015-012†

October 5, 2015, revised January 6, 2016 and April 6, 2016,
accepted for publication in **European Journal of Operational Research**

Abstract

The starting point used by an interior point algorithm for linear and convex quadratic programming may significantly influence the behaviour of the method. A widely used heuristic to construct such a point consists of dropping variable nonnegativity constraints and computing a solution which minimizes the Euclidean norm of the primal (or dual) point while satisfying the appropriate primal (or dual) equality constraints, followed by shifting the variables so that all their components are positive and bounded away from zero. In this Short Communication a new approach for finding a starting point is proposed. It relies on a few inexact Newton steps performed at the start of the solution process. A formal justification of the new heuristic is given and computational results are presented to demonstrate its advantages in practice. Computational experience with a large collection of small- and medium-size test problems reveals that the new starting point is superior to the old one and saves 20-40% of iterations needed by the primal-dual method. For larger and more difficult problems this translates into remarkable savings in the solution time.

Keywords: Interior Point Methods, Initial Point, Crash Start, Linear Programming, Quadratic Programming.

*Email: J.Gondzio@ed.ac.uk, URL: http://maths.ed.ac.uk/~gondzio/
†For other papers in this series see http://www.maths.ed.ac.uk/ERGO/

1 Introduction

We are concerned in this paper with the efficient solution of linear and convex quadratic programming problems using interior point methods (IPMs). Such problems are at the heart of many more complicated optimization techniques and progress in their solution impacts the whole optimization area. Following the notation of [9] we consider the following general primal-dual pair of convex quadratic programming (QP) problems

$$\begin{array}{lll} \text{Primal} & & \text{Dual} \\ \min & c^Tx + \frac{1}{2}x^TQx & \max \quad b^Ty - \frac{1}{2}x^TQx \\ \text{s.t.} & Ax = b, & \text{s.t.} \quad A^Ty + s - Qx = c, \\ & x \geq 0; & \quad\; y \text{ free}, \;\; s \geq 0, \end{array} \tag{1}$$

where $A \in \mathcal{R}^{m\times n}$ has full row rank $m \leq n$, $Q \in \mathcal{R}^{n\times n}$ is a positive semidefinite matrix, $x, s, c \in \mathcal{R}^n$ and $y, b \in \mathcal{R}^m$. In the special case when $Q = 0$ the problems become the pair of primal-dual linear programming (LP) problems.

Standard interior point methods are very sensitive to the choice of a starting point. Many codes use an idea of Mehrotra [17] and construct a point by solving an auxiliary quadratic programming problem: $\min c^Tx + \frac{1}{2}x^TQx + \frac{1}{2}x^Tx$ s.t. $Ax = b$. In this problem all equality constraints are satisfied, but the simple inequalities are ignored. The solution of such a problem may be obtained by an explicit formula at a cost comparable to a single interior point iteration. Since the non-negativity constraint $x \geq 0$ is dropped in it, the solution might contain negative components. Therefore to become an eligible starting point for an IPM, they need to be shifted to positive values. A similar auxiliary problem is formulated to determine an initial dual solution (y, s). Several attempts have been made to improve on this (heuristic) starting point selection and, although some of them offered attractive alternative initialization methods for particular classes of problems, to the best of the author's knowledge, they do not offer a competitive approach for the general case. It is worth mentioning that if a self-dual emdedding [21] is used then it is possible to accommodate an arbitrary point and convert it into a starting point [19]. However, the implementation of self-dual emdedding needs a slightly more involved linear algebra step (one more back-solve per iteration) and we are not going to use it here.

In this paper we propose a new approach which finds a good initial point for interior point methods applied to a *general* convex quadratic programming problem. We call it a *crash start technique* because it follows a similar principle to that employed by simplex solvers and attempts to guess a starting point which is closer to optimality than a default one. Crash start has proved very useful in the context of simplex method for linear programming [2, 13, 14, 16]. Such procedures usually look for an advanced initial basis in which columns corresponding to slack and artificial variables are replaced by those corresponding to structural variables. The heuristics are based on a general expectation that the more those structural columns are inserted into the initial basis the closer it might be to the optimal basis.

The situation in interior point methods is significantly more complicated. Modern IPMs owe their efficiency to the ability to follow closely the central path [9, 20]. Indeed, both the theory and the computational practice confirm that, as long as the iterates remain in the proximity of the central path, fast progress to optimality can be made. Conversely, if the iterates leave the vicinity of the central path and prematurely approach the boundary of the feasible region, the algorithm might get stuck taking small steps in the Newton direction and the convergence might be disappointingly slow. This means in particular that IPMs cannot be started successfully from an arbitrary point. An ideal initial point should satisfy several requirements:

- it should be close to primal and dual feasibility;
- it should be well centred;
- it should be as close to optimality as possible.

Finding such a point is by no means easy!

In this paper we propose a practical method to construct a point which satisfies all three requirements. Recently there has been a major increase in interest in the use of iterative methods to compute Newton directions in IPMs [5, 9] and a variety of preconditioners for Krylov subspace methods applied in this context have been proposed. Many preconditioners have already been proposed for the normal equations (Schur complement of the KKT system) [3, 4, 18] as well as for the indefinite augmented form of the KKT system [7, 8, 15]. There is increasing evidence that using inexact Newton directions [6] in interior point methods is well supported by the theory [1, 11] and works well in practice [9]. Our *crash start technique* builds upon these developments.

We observe that at the beginning of the solution process an infeasible interior point method works with large infeasibilities in the primal and dual spaces and a large duality gap. Therefore computing highly accurate Newton directions is not necessary at this stage; very crude inexact directions are able to offer noticeable progress in reducing infeasibilities and the duality gap. Such inexact directions can be computed at a significantly lower cost than exact ones. Consequently we propose to run several initial iterations with directions computed by a preconditioned Krylov subspace method using a very simple (and inexpensive) preconditioner and asking only for very relaxed accuracy to make sure that a few Krylov iterations are enough to deliver an inexact solution. Our choice is a partial Cholesky preconditioner which was designed specially for the matrix-free IPM [10]. This preconditioner has several advantages including simplicity and ability to work with very limited (and easy to control) memory requirements.

In our developments in this Short Communication we will follow very closely the recent EJOR survey [9] and therefore we will focus only on several computational aspects which are relevant to the understanding of our crash start approach. Hence this short paper has the following simple structure. In Section 2 we will present the key ideas of interior point methods and in Section 3 we will discuss in detail our new crash start procedure and its implementation. Although our approach to generate an advanced initial solution is only a heuristic, we will provide some simple theoretical justification for it. In Section 4 we will present a comparison of two variants of the interior point method, one using a standard default starting point and one initialized with the proposed crash start solution. Finally, in Section 5 we will give our conclusions.

2 Basics of Interior Point Methods

Path-following interior point methods are well-understood [9, 20] and very powerful optimization techniques. An IPM for quadratic programming may be interpreted as an iterative method which follows the path of solutions of the following perturbed first order optimality conditions for (1)

$$\begin{aligned} Ax &= b, \\ A^T y + s - Qx &= c, \\ XSe &= \mu e, \\ (x,s) &\geq 0. \end{aligned} \tag{2}$$

We use a standard IPM notation in which X and S are diagonal matrices in $\mathcal{R}^{n\times n}$ with elements of vectors x and s spread across the diagonal, respectively and $e \in \mathcal{R}^n$ is the vector of ones.

IPMs use the notion of a primal-dual *central path*, being the set of solutions of (2) for any $\mu > 0$. It can be shown that the set of such solutions forms a continuous path $\{(x(\mu), y(\mu), s(\mu)) : \mu > 0\}$ and much evidence has been gathered to date that interior point methods benefit from following this path closely [12]. In this paper we consider a primal-dual infeasible IPM and therefore define the *symmetric primal-dual infeasible neighbourhood* of the central path as follows

$$N_S(\gamma, \beta) = \{(x, y, s) \,|\, \|\xi_p\| \leq \frac{\beta\mu}{\mu^0}\|\xi_p^0\|,\ \|\xi_d\| \leq \frac{\beta\mu}{\mu^0}\|\xi_d^0\|,\ \gamma\mu \leq x_j s_j \leq \frac{1}{\gamma}\mu\}, \tag{3}$$

where $\xi_p = b - Ax$ and $\xi_d = c - A^T y - s + Qx$ are the violations of primal and dual feasibility constraints, respectively, the superscript zero denotes the initial values of the barrier parameter μ and the infeasibilities ξ_p and ξ_d, $\gamma \in (0, 1)$ controls the width of the neighbourhood and β is a constant. From a computational perspective the most demanding task in IPMs is the computation of the Newton direction $(\Delta x, \Delta y, \Delta s)$ for the nonlinear system (2) which requires solving the following system of linear equations

$$\begin{bmatrix} A & 0 & 0 \\ -Q & A^T & I \\ S & 0 & X \end{bmatrix} \cdot \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta s \end{bmatrix} = \begin{bmatrix} \xi_p \\ \xi_d \\ \xi_\mu \end{bmatrix} = \begin{bmatrix} b - Ax \\ c - A^T y - s + Qx \\ \sigma\mu e - XSe \end{bmatrix}. \tag{4}$$

The parameter $\sigma \in (0, 1)$ controls the aspiration of how much one would like to reduce the barrier term $\mu^{k+1} = \sigma\mu^k$. Given the Newton direction, a maximum stepsize α which keeps the new iterate $(\bar{x}, \bar{y}, \bar{s}) = (x, y, s) + \alpha(\Delta x, \Delta y, \Delta s)$ in the neighbourhood (3) is determined and then the algorithm makes this step to a new iterate. We summarize the algorithm below.

INFEASIBLE PATH-FOLLOWING METHOD FOR QUADRATIC PROGRAMMING

Parameters

$\sigma \in (0, 1)$ barrier reduction parameter;
$\gamma = 0.1,\ \beta = 2$ parameters of the infeasible neighbourhood;
ε_p, ε_d, ε_o primal feasibility, dual feasibility and optimality tolerances:
IPM stops when $\frac{\|\xi_p^k\|}{\|\xi_p^0\|} \leq \varepsilon_p$, $\frac{\|\xi_d^k\|}{\|\xi_0^0\|} \leq \varepsilon_d$ and $\frac{(x^k)^T s^k/n}{1+|c^T x^k + 1/2(x^k)^T Q x^k|} \leq \varepsilon_o$.

Initialize IPM

iteration counter $k = 0$;
primal-dual point $(x^0, y^0, s^0) \in N_S(\gamma, \beta)$;
barrier parameter $\mu^0 = (x^0)^T s^0/n$;
primal and dual infeasibilities $\xi_p^0 = b - Ax^0$ and $\xi_d^0 = c - A^T y^0 - s^0 + Qx^0$.

Interior Point Method

while ($\frac{\|\xi_p^k\|}{\|\xi_p^0\|} > \varepsilon_p$ or $\frac{\|\xi_d^k\|}{\|\xi_d^0\|} > \varepsilon_d$ or $\frac{(x^k)^T s^k/n}{1+|c^T x^k + 1/2(x^k)^T Q x^k|} > \varepsilon_o$) **do**

Update (reduce) the barrier $\mu^{k+1} = \sigma\mu^k$;

Solve the KKT system (4): find the primal-dual Newton direction $(\Delta x, \Delta y, \Delta s)$.

Find $\alpha = \max\{\alpha : (x^k + \alpha\Delta x,\ y^k + \alpha\Delta y,\ s^k + \alpha\Delta s) \in N_S(\gamma, \beta)\}$;
Make step
$(x^{k+1}, y^{k+1}, s^{k+1}) = (x^k + \alpha\Delta x, y^k + \alpha\Delta y, s^k + \alpha\Delta s)$.
Compute the infeasibilities: $\xi_p^{k+1} = b - Ax^{k+1}$ and $\xi_d^{k+1} = c - A^T y^{k+1} - s^{k+1} + Qx^{k+1}$;
Update the iteration counter: $k := k + 1$.

end-while

The theory of IPMs requires a tight control of the speed at which the sequence $\{\mu^k\}$ converges to zero. The right balance needs to be struck between keeping the iterates (x, y, s) in the interior, that is satisfying $XSe \approx \mu e$ with $x > 0$ and $s > 0$, and forcing progress to optimality. Hence to guarantee convergence IPMs gradually shrink the barrier term μ. If the iterates stay in the neighbourhood (3) then the primal and dual infeasibilities are reduced at the same speed as the barrier parameter μ. When μ gets sufficiently close to zero then, by virtue of (3), the violations in all three equations in (2) get small and the algorithm terminates by finding an (approximate) optimal solution.

We skip any detailed convergence analysis of the method and refer the interested reader to [9, 20]. Instead, in the next section we will draw reader's attention to several interesting features of interior point methods which will then be exploited to construct an advanced initial point. In particular, it is worth having a closer look at the structure of the first order optimality conditions (2) and the corresponding Newton system (4).

3 Crash Start Procedure and its Implementation

The first two equations in (2) which correspond to primal and dual feasibility conditions in (1) are linear. Using simple linear algebra manipulations it is easy to show that after a step to a new point is made in the Newton direction (4) the new primal and dual infeasibilities satisfy $\bar{\xi}_p = (1-\alpha)\xi_p$ and $\bar{\xi}_d = (1-\alpha)\xi_d$. Hence as long as the stepsize α is bounded away from zero the progress in reducing infeasibilities is very fast. The third equation in (2) corresponds to the perturbed complementarity condition and is (mildly) nonlinear, bilinear in fact. Consequently, the third equation in (4) displays a "symmetric" structure $S\Delta x + X\Delta s = \xi_\mu$ and proves very handy when one evaluates what happens to the complementarity product after a step is taken in the Newton direction. The gap at the new point becomes

$$\bar{x}^T\bar{s} \;=\; (x+\alpha\Delta x)^T(s+\alpha\Delta s) \;=\; x^Ts + \alpha(s^T\Delta x + x^T\Delta s) + \alpha^2\Delta x^T\Delta s. \tag{5}$$

Only the term $\Delta x^T\Delta s$ which involves the second-order error remains difficult to handle and, indeed, much of the effort of any theoretical analysis of IPMs focuses on obtaining bounds on this term. If the second order term $\Delta x^T\Delta s$ is kept small then significant progress can be made in reducing the complementarity gap when making a step in the Newton direction.

IPMs work with perturbed optimality conditions and the barrier term μ in (4) is systematically reduced by a factor of σ at each iteration. Therefore, for the efficiency of interior point algorithms it is not necessary to solve this system of nonlinear equations to a high degree of accuracy. Recall that (2) is only an approximation of the optimality conditions for (1) corresponding to a specific choice of the barrier parameter μ and the barrier term will have to be reduced further to force convergence.

Rather than solving (4) exactly, we will assume that the system is solved only approximately, that is, an *inexact Newton direction* $(\Delta\tilde{x}, \Delta\tilde{y}, \Delta\tilde{s})$ is computed which satisfies

$$\begin{bmatrix} A & 0 & 0 \\ -Q & A^T & I \\ S & 0 & X \end{bmatrix} \cdot \begin{bmatrix} \Delta\tilde{x} \\ \Delta\tilde{y} \\ \Delta\tilde{s} \end{bmatrix} = \begin{bmatrix} \xi_p \\ \xi_d \\ \xi_\mu \end{bmatrix} + \begin{bmatrix} r_p \\ r_d \\ r_\mu \end{bmatrix}, \tag{6}$$

where r_p and r_d denote errors in the primal and dual feasibility constraints, respectively and r_μ is the error in complementarity condition. We observe the following important property.

Observation 3.1 *Let $\delta \in [0,1)$ be given. Assume an inexact solution $(\Delta\tilde{x}, \Delta\tilde{y}, \Delta\tilde{s})$ of (6) satisfies*

$$\|r_p\| \leq \delta\|\xi_p\|, \quad \|r_d\| \leq \delta\|\xi_d\| \quad \textit{and} \quad \|r_\mu\| \leq \delta\|\xi_\mu\| \tag{7}$$

and a step in this direction is taken with a maximum stepsize α which keeps the new iterate

$$(\tilde{x}, \tilde{y}, \tilde{s}) = (x, y, s) + \alpha(\Delta\tilde{x}, \Delta\tilde{y}, \Delta\tilde{s}) \tag{8}$$

in the infeasible symmetric neighbourhood (3). Then the new primal and dual infeasibilities satisfy

$$\|\tilde{\xi}_p\| \leq (1 - \alpha(1-\delta))\|\xi_p\| \quad \textit{and} \quad \|\tilde{\xi}_d\| \leq (1 - \alpha(1-\delta))\|\xi_d\|. \tag{9}$$

It is worth comparing what happens to infeasibilities when a step in the *exact* Newton direction (4) is taken and when a step in the *inexact* Newton direction (6) is made. The use of exact directions guarantees a fast reduction of infeasibilities as long as large stepsizes α are used. One would wish the stepsizes to be as close to 1 as possible. If at any iteration $\alpha = 1$ and a full Newton step is made then the infeasibilities are reduced to zero and are guaranteed to remain zero until termination of the algorithm. The use of inexact directions might slow down the reduction of infeasibilities, but not so much. For example if $\delta \leq 0.5$ then $\tilde{\xi}_p$ and $\tilde{\xi}_d$ are still guaranteed to be reduced by a factor of at least $1 - 0.5\alpha$. Considering that the first two equations in (2) are linear, the fast reduction of errors in these equations is observed regardless of whether exact or inexact Newton directions are used.

It is also interesting to analyze how the use of inexact directions affects the reduction of the complementarity product (5). The gap at the new point (8) becomes

$$\tilde{x}^T\tilde{s} \;=\; (x + \alpha\Delta\tilde{x})^T(s + \alpha\Delta\tilde{s}) \;=\; x^Ts + \alpha(s^T\Delta\tilde{x} + x^T\Delta\tilde{s}) + \alpha^2\Delta\tilde{x}^T\Delta\tilde{s}. \tag{10}$$

The second term in the summation is determined by the third equation in (6) hence

$$s^T\Delta\tilde{x} + x^T\Delta\tilde{s} = e^T\xi_\mu + e^Tr_\mu. \tag{11}$$

In this case, an assumption similar to (7) which controls the norm of the error r_μ is too general to derive a rigorous bound which would guarantee that (11) forces sufficient reduction of the complementarity gap. We would need a stronger condition, for example,

$$e^Tr_\mu \leq -\delta e^T\xi_\mu. \tag{12}$$

Such a requirement guarantees that the presence of the error r_μ worsens the "optimizing" properties of inexact Newton direction only by a fraction. This is clearly an implementable condition which can be verified while using any iterative method to solve (6).

A rigorous analysis of such a situation would also have to deliver bounds on the second-order term $\Delta\tilde{x}^T\Delta\tilde{s}$ to guarantee that this error is small in comparison with the other terms in (10). Such analysis is very difficult. Besides we are not going to use inexact method to reach convergence. Instead of that, we will use the inexact IPM only to provide an advanced starting point for the standard (exact) method, i.e., we focus on providing a heuristic to generate a crash start point.

In summary, our crash start routine runs a few iterations of the inexact infeasible primal-dual path-following algorithm. The method differs from an exact algorithm presented at the end of Section 2 by using inexact directions obtained by solving (6) (instead of (4)). The accuracy requirements (7) and (12) are imposed to guarantee sufficient progress of the algorithm. For the success of our approach it is crucial to be able to solve equations (6) very fast. We expect to achieve it by (i) using an iterative scheme with an inexpensive preconditioner, and (ii) asking

only for a loose accuracy δ in (7) and (12). This will be delivered by applying the partial Cholesky preconditioner used in the matrix-free interior point method [10]. We refer the reader to [9, 10] for more detail.

The Newton equation system—whether exact (4) or inexact (6)—involves the same highly structured block 3x3 matrix. It is usually reduced to a 2x2 (or a 1x1) system by eliminating Δs (or both Δs and Δx). In a standard IPM such systems are usually solved by a direct approach. In the approach proposed in this paper we apply the preconditioned conjugate gradient method to solve the 1x1 system resulting from the reduction of (6). Below we describe in more detail the proposed crash start technique.

We set very loose requirements on the accuracy of Newton directions, for example, by choosing $\delta = 0.1$ in (7) and run a few very inexpensive interior point iterations until the infeasibilities in the primal and dual spaces and the duality gap are noticeably reduced compared with their initial values. This corresponds to running a path-following algorithm similar to that presented at the end of Section 2 with the following primal feasibility, dual feasibility and optimality tolerances

$$\varepsilon_p = 10^{-3}, \quad \varepsilon_d = 10^{-3} \quad \text{and} \quad \varepsilon_o = 10^{-1}, \tag{13}$$

respectively. Inexact Newton directions are computed by reducing (6) to the normal equations form and solving it using the special preconditioned conjugate gradient (PCG) method which employs the partial Cholesky preconditioner [10] of the normal equation matrix. The preconditioner is allowed to use at most k_{max} columns in the partial Cholesky. We set the accuracy requirement in PCG to $\varepsilon_{PCG} = 10^{-3}$, however, to avoid excessively long runs we additionally limit the number of PCG iterations at a user-defined maximum, PCG_{max}. In fact, we are ready to accept quite inaccurate directions following the logic justified by Observation 3.1 that even such inexact directions guarantee good progress in reducing primal and dual infeasibilities.

When the inexact infeasible algorithm terminates by satisfying conditions (13), the last IPM iterate (x^k, y^k, s^k) is passed as a *crash start* point to a standard primal-dual path-following interior point algorithm which then finds the optimal solution to high tolerances.

It is worth adding that it is possible to run the matrix-free method with more demanding stopping criteria, that is with smaller feasibility and optimality tolerances. However, this would require increasing the rank of the partial Cholesky factorization to produce a better preconditioner and allowing it to perform more conjugate gradient iterations to reduce the error in the inexact Newton directions.

4 Computational Results

The crash start procedure proposed in this paper has been implemented in the author's research interior point solver HOPDM [9]. Below we compare the crash start version against the standard HOPDM solver by applying both to solve a large set of linear and quadratic programming problems. All our tests have been performed on a MacBook Pro with Intel Core i7 processor running at 2.3 GHz with 6MB level 3 cache and 16GB of RAM.

In our first experiment, we have tested the approach on a collection of 111 small- to medium-scale linear programs: 95 problems available from netlib `http://www.netlib.org/lp/data/` and 16 problems of Kennington `http://www.netlib.org/lp/data/kennington/`. The sizes of netlib problems vary from a few hundred to a few thousand constraints and variables and

therefore many of these problems are solved in a small fraction of a second. The sizes of network problems in the Kennington collection are larger: there is one problem with more than 100,000 constraints, several problems have a few hundred thousand variables and just one problem with the number of nonzero entries in the matrix A exceeding 1,000,000. The solution of each of these problems usually takes a few seconds. These two collections still provide a reasonable set of test examples and it is worth checking how a new method performs on them.

To evaluate the performance of the crash start heuristic we have applied it to all problems from these two collections. We run the matrix-free method until conditions (13) are satisfied and then switch to standard IPM which uses exact Newton directions. The matrix-free method is run with the partial Cholesky preconditioner limited to $k_{max} = 100$ columns and the preconditioned conjugate gradient method attempts to achieve the relative accuracy $\varepsilon_{PCG} = 10^{-6}$, but if it struggles with reaching this accuracy then it is interrupted after at most $PCG_{max} = 100$ iterations.

The aggregated results of these runs for both test collections are presented under the heading "Simple Crash" in columns 5–8 of Table 1. We report the number of crash iterations (iC) and the associated time spent in constructing crash start point (tC), followed with the performance measures (its and time) of the standard IPM iterations executed to reach optimality. As a reference, in columns 3 and 4 of Table 1, we report the number of IPM iterations and the CPU time when the standard primal-dual code HOPDM [9] is used, employing a direct solver to compute the exact Newton direction at each iteration.

Problem Set	No of problems	HOPDM		Simple Crash			
		its	time	iC	tC	its	time
Netlib	95	1297	30.70	603	11.59	789	21.74
Kennington	16	297	157.91	93	18.49	219	118.38
Totals	111	1594	188.61	696	30.08	1008	140.12

Table 1: Crash start results for netlib and Kennington problems.

Observe that the use of inexact Newton directions in the first phase of optimization only marginally increases the overall number of interior point iterations. The total numbers of iterations needed to solve all 111 problems are reported in the last row of Table 1. It has grown from 1594 (standard HOPDM) to 696+1008 = 1704, a growth of merely 7%. However, the use of the crash start has reduced the number of *expensive* IPM iterations from 1594 to 1008 which is a reduction of nearly 37%. Naturally the reduction of such iterations usually translates into CPU time savings and these are more significant for problems which need an expensive Cholesky factorization. Indeed, the solution time of all 111 problems has been reduced from 188.61s to 30.08+140.12 = 170.20s, which is about 10% saving. Observe that if the problem is very easy and the Cholesky factor of the normal equations matrix is very sparse (as is the case of many linear programming problems) then an IPM iteration which computes the exact direction by using a direct solver is hard to compete with. In such case an inexact solution of Newton system which performs several CG iterations each involving multiplications with matrix A and its transpose is simply too expensive. This is clearly seen in case of solving the netlib problems for which the overall solution time has increased from 30.70s to 11.59+21.74 = 33.33s.

We illustrate this by reporting in Table 2 more details of the runs for a small but representative subset of linear programs. For three problems from the Netlib collection and three problems from the Kennington collection we report the number of nonzero entries in the matrix A and in the Cholesky factor L of the normal equation matrix as well as the number of iterations and CPU time in seconds for the standard primal-dual interior point method, HOPDM. Then in

the last six columns of Table 2 we report the largest recorded number of nonzero entries in the partial Cholesky preconditioner nz(P), the largest recorded number of CG iterations and, finally, the numbers of crash iterations (iC) and exact IPM iterations (its) along with the CPU times in seconds (tC and time, respectively).

Problem	HOPDM				Simple Crash					
	nz(A)	nz(L)	its	time	nz(P)	CG	iC	tC	its	time
pilot	43220	231045	17	0.65	67756	100	11	0.83	11	0.40
pilot87	73804	428310	16	1.70	93573	100	8	0.96	11	1.08
stocfor3	74004	264372	21	0.76	2294	100	7	1.03	16	0.54
ken-13	139834	204506	15	0.74	21261	39	2	0.20	13	0.59
pds-10	140063	1594136	18	12.56	10414	59	5	0.75	15	10.18
pds-20	304153	6678041	21	116.50	3646	100	6	2.79	17	92.93

Table 2: Simple Crash: Sparsity of factors and CPU times.

The analysis of these results confirms that if too much effort is spent in the conjugate gradient algorithm to compute inexact Newton directions then the CPU time to construct the crash start point exceeds the savings resulting from doing fewer exact IPM iterations. This is not a surprise; similar observations were made when the matrix-free method [9, 10] was applied to other sparse problems with inexpensive Cholesky factors of the normal equation matrix. Indeed, unless the cost of computing this Cholesky factor is much higher than the cost of matrix-vector multiplications with A and its transpose (required at each CG iteration), it is simpler and more efficient to compute the exact Newton directions. This is demonstrated for example in the time needed to perform 7 matrix-free iterations when solving `stocfor3`. To execute 100 CG iterations, 200×74004 floating point operations are needed to do matrix-vector products alone and this significantly exceeds the effort to compute and apply the Cholesky factor which has 264372 nonzero entries.

These observations have led us to propose a significantly more aggressive strategy for computing the crash start point. To avoid long runs of the iterative method when solving Newton equations, we set the maximum number of inexact iterations to 6 and the maximum number of CG iterations when solving any linear equation to $PCG_{max} = 6$. The results of runs of such a (better) crash start strategy are presented in Table 3. The restrictions imposed on numbers of inexact iterations and CG steps cut down the time spent to compute a crash start point significantly. Although the numbers of exact IPM iterations might have increased slightly compared with the simple crash (Table 2), this new strategy is more likely to guarantee time savings over the default exact method.

Problem	HOPDM				Better Crash					
	nz(A)	nz(L)	its	time	nz(P)	CG	iC	tC	its	time
pilot	43220	231045	17	0.65	70209	6	6	0.09	14	0.50
pilot87	73804	428310	16	1.70	94889	6	6	0.13	15	1.51
stocfor3	74004	264372	21	0.76	2294	6	6	0.18	17	0.60
ken-13	139834	204506	15	0.74	23389	6	5	0.26	10	0.44
pds-10	140063	1594136	18	12.56	5656	6	6	0.31	17	11.64
pds-20	304153	6678041	21	116.50	2445	6	6	0.73	15	81.23

Table 3: Better Crash: Sparsity of factors and CPU times.

In our next experiment the same crash start is applied to solve larger linear programs made available by Dr C. Meszaros at `http://www.sztaki.hu/~meszaros/public_ftp/lptestset/`, used

also in Prof. H. Mittelmann's benchmarks of LP solvers `http://plato.asu.edu/bench.html`. In Table 4 we report problem sizes. Some of the problems have several hundred thousand constraints, more than a million of variables and tens of million nonzero entries in A. Then we report the solution statistics for the standard primal-dual interior point method, HOPDM: the number of nonzero entries in the Cholesky factor L of a 2x2 or 1x1 system (nz(L)), the number of IPM iterations and CPU time in seconds. Finally in the last five columns we report the largest recorded number of nonzero entries in the partial Cholesky preconditioner nz(P), the numbers of crash iterations (iC) and exact IPM iterations (its) along with the CPU times in seconds (tC and time, respectively).

Problem	Dimensions			HOPDM			Better Crash				
	rows	columns	nonzeros	nz(L)	its	time	nz(P)	iC	tC	its	time
dbic1	43200	183235	1217046	1827660	26	14.14	288104	6	1.71	21	10.96
degme	185501	659415	8714608	55486470	17	2299.81	18269758	6	41.62	16	2032.17
karted	46502	133115	1888729	38743737	10	841.61	4643676	6	8.44	9	700.28
neos	479119	36786	1084461	9719640	40	204.96	785236	6	6.48	32	158.61
rail4284	4284	1092610	12372358	5706959	21	198.99	302302	6	10.01	18	164.30
sgpf5y6	246077	308634	902275	1260077	22	8.24	2170	6	1.86	20	6.94
spal-004	10203	321696	46167727	43432068	20	2885.96	878200	5	11.69	19	2651.21
storm1000	528185	1259121	4228817	10698471	62	308.52	587	6	10.20	57	276.41
tp-6	142752	1014301	12440225	29154387	19	1074.00	14232942	6	42.81	16	859.22
ts-palko	22002	47235	1119043	15482659	11	206.49	2194964	6	3.33	11	190.19

Table 4: Crash start results for large linear programs.

The analysis of these results reveals that for more difficult linear programs crash start is an attractive technique which reduces the overall time needed to solve the problems. To avoid obfuscating the results, all runs reported in Table 4 have been obtained with the same rule to decide when a switch from crash to standard (exact) iterations is made as those reported in Tables 3 and 6. However, the analysis of these results indicates that it might be useful to perform a fine-tuning of the heuristic and allow such a switching between crash and standard iterations to become problem-dependent. Following the referee's suggestion, we have implemented a simple dynamic switching mechanism. We compare the flop estimates of the last inexact iteration C_i with that of exact iteration C_e and as soon as $C_i > 0.5C_e$ we switch to exact algorithm. The estimates of costs are

$$\begin{aligned} C_i &= flops(P) + its_{CG} \times (nz(P) + 2nz(A)) \\ C_e &= flops(L) + 2 \times (nz(L) + 2nz(A)), \end{aligned}$$

where $flops(P)$ and $flops(L)$ are flop counts to compute the partial Cholesky preconditioner and the Cholesky factor, respectively, and $nz(K)$ denotes the number of nonzero entries in matrix K. We also set $\varepsilon_o = 10^{-3}$ in (13) to allow more inexact iterations if those are inexpensive. We have repeated test runs for all large linear programs and report the results in Table 5. We observe that a dynamic switch from inexact to exact iterations usually improves the solution time.

Problem	Better Crash				Crash with Fallback			
	iC	tC	its	time	iC	tC	its	time
dbic1	6	1.71	21	10.96	8	1.99	20	10.46
degme	6	41.62	16	2032.17	6	41.62	16	2032.17
karted	6	8.44	9	700.28	9	12.70	8	639.15
neos	6	6.48	32	158.61	7	7.33	34	168.61
rail4284	6	10.01	18	164.30	5	8.13	18	164.74
sgpf5y6	6	1.86	20	6.94	7	2.17	18	6.37
spal-004	5	11.69	19	2651.21	8	17.22	18	2544.18

Table 5: Crash start results for large linear programs.

Problem	Better Crash				Crash with Fallback			
	iC	tC	its	time	iC	tC	its	time
storm1000	6	10.20	57	276.41	11	17.88	52	251.14
tp-6	6	42.81	16	859.22	7	50.19	15	801.46
ts-palko	6	3.33	11	190.19	10	5.26	10	174.39

Table 5: Crash start results for large linear programs.

In our final experiment we run the same crash start routine on medium-scale quadratic programming problems made available by Prof. H. Mittelmann. The interested reader may generate them using AMPL files from: `ftp://plato.la.asu.edu/pub/vlgqp.txt`. In Table 6 we compare the performance of our better crash start routine and the standard primal-dual solver HOPDM [9] using its default settings. The maximum number of inexact iterations is set to 6 and the maximum number of CG iterations at any stage is also limited to 6. For standard HOPDM runs we report the number of nonzeros in the Cholesky factor L of a 2x2 or 1x1 system, the number of IPM iterations and the CPU time in seconds. For crash start runs we report the largest recorded number of nonzero entries in the partial Cholesky preconditioner nz(P), the numbers of crash iterations and exact IPM iterations along with the CPU times in seconds.

Problem	HOPDM			Better Crash				
	nz(L)	its	time	nz(P)	iC	tC	its	time
aug2dcqp	456846	8	0.39	1105	5	0.20	3	0.10
aug2dqp	444184	9	0.34	1147	4	0.14	5	0.14
aug3d	2446246	9	10.41	763	6	0.37	8	8.28
aug3dc	5269968	19	61.70	677	6	0.68	17	54.57
aug3dcqp	5336157	17	54.97	665	6	0.50	15	49.03
aug3dqp	1507295	10	2.70	1634	4	0.13	6	0.92
sqp2500-1	1909672	14	17.16	179692	6	0.22	11	14.07
sqp2500-2	1909275	13	14.61	177263	6	0.21	10	12.44
sqp2500-3	9874267	13	164.07	410575	6	0.57	10	121.21

Table 6: Crash start results for quadratic programs.

5 Conclusions

A new strategy to construct a crash start solution for interior point methods has been proposed in this paper. It exploits the inherent ability of IPMs to make very fast progress at the beginning of optimization by using inexact (hence inexpensive to compute) Newton directions. A brief theoretical justification of the crash start strategy has been provided and a practical algorithm has been demonstrated to deliver noticeable improvement over a standard implementation of the interior point method.

Acknowledgements
The author is grateful to Dr Julian Hall for reading the paper and providing useful suggestions which led to an improvement of the presentation. The author is also grateful to the anonymous referees for their comments which helped to improve the paper.

References

[1] S. Bellavia, *Inexact interior point method*, Journal of Optimization Theory and Applications, 96 (1998), pp. 109–121.

[2] R. E. Bixby, *Implementing the simplex method: the initial basis*, ORSA Journal on Computing, 4 (1992), pp. 267–284.

[3] S. Bocanegra, F. Campos, and A. Oliveira, *Using a hybrid preconditioner for solving large-scale linear systems arising from interior point methods*, Computational Optimization and Applications, 36 (2007), pp. 149–164.

[4] J. Castro, *A specialized interior-point algorithm for multicommodity network flows*, SIAM Journal on Optimization, 10 (2000), pp. 852–877.

[5] M. D'Apuzzo, V. De Simone, and D. di Serafino, *On mutual impact of numerical linear algebra and large-scale optimization with focus on interior point methods*, Computational Optimization and Applications, 45 (2010), pp. 283–310.

[6] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM Journal on Numerical Analysis, 19 (1982), pp. 400–408.

[7] C. Durazzi and V. Ruggiero, *Indefinitely preconditioned conjugate gradient method for large sparse equality and inequality constrained quadratic problems*, Numerical Linear Algebra with Applications, 10 (2003), pp. 673–688.

[8] P. E. Gill, W. Murray, D. B. Ponceleón, and M. A. Saunders, *Preconditioners for indefinite systems arising in optimization*, SIAM Journal on Matrix Analysis and Applications, 13 (1992), pp. 292–311.

[9] J. Gondzio, *Interior point methods 25 years later*, European Journal of Operational Research, 218 (2012), pp. 587–601.

[10] ——, *Matrix-free interior point method*, Computational Optimization and Applications, 51 (2012), pp. 457–480.

[11] ——, *Convergence analysis of an inexact feasible interior point method for convex quadratic programming*, SIAM Journal on Optimization, 23 (2013), pp. 1510–1527.

[12] C. C. Gonzaga, *Path–following methods for linear programming*, SIAM Review, 34 (1992), pp. 167–224.

[13] N. I. M. Gould and J. K. Reid, *New crash procedures for large-scale systems of linear constraints*, Mathematical Programming, 45 (1989), pp. 475–501.

[14] J. A. J. Hall, *Towards a practical parallelisation of the simplex method*, Computational Management Science, 7 (2010), pp. 139–170.

[15] L. Lukšan and J. Vlček, *Indefinitely preconditioned inexact Newton method for large sparse equality constrained nonlinear programming problems*, Numerical Linear Algebra with Applications, 5 (1998), pp. 219–247.

[16] I. Maros and G. Mitra, *Startegies for creating advanced bases for large-scale linear programming problems*, INFORMS Journal on Computing, 10 (Spring 1998), pp. 248–260.

[17] S. Mehrotra, *On the implementation of a primal–dual interior point method*, SIAM Journal on Optimization, 2 (1992), pp. 575–601.

[18] A. R. L. Oliveira and D. C. Sorensen, *A new class of preconditioners for large-scale linear systems from interior point methods for linear programming*, Linear Algebra and its Applications, 394 (2005), pp. 1–24.

[19] A. Skajaa, E. Andersen, and Y. Ye, *Warmstarting the homogeneous and self-dual interior point method for linear and conic quadratic problems*, Mathematical Programming Computation, 5 (2013), pp. 1–25.

[20] S. J. Wright, *Primal-Dual Interior-Point Methods*, SIAM, Philadelphia, 1997.

[21] Y. Ye, M. Todd, and S. Mizuno, *An $o(\sqrt{n}L)$ - iteration homogeneous and self-dual linear programming algorithm*, Mathematics of Operations Research, 19 (1994), pp. 53–67.