Large-Scale Convex Optimization via Saddle Point Computation

Kallio, M.J. & Rosa, C.H.

IIASA Working Paper

WP-94-107

October 1994

Large-Scale Convex Optimization via Saddle Point Computation

Markku Kallio
Charles H. Rosa

WP-94-107
October 1994

Large-Scale Convex Optimization via Saddle Point Computation

Markku Kallio
Charles H. Rosa

WP-94-107
October 1994

Abstract

This article proposes large-scale convex optimization problems to be solved via saddle points of the standard Lagrangian. A recent approach for saddle point computation is specialized, by way of a specific perturbation technique and unique scaling method, to convex optimization problems with differentiable objective and constraint functions. In each iteration the update directions for primal and dual variables are determined by gradients of the Lagrangian. These gradients are evaluated at perturbed points which are generated from current points via auxiliary mappings. The resulting algorithm suits massively parallel computing. Sparsity can be exploited efficiently. Employing simulation of parallel computations, an experimental code embedded into GAMS is tested on two sets of nonlinear problems. The first set arises from multi-stage stochastic optimization of the US energy economy. The second set consists of multi-currency bond portfolio problems. In such stochastic optimization problems the serial time appears approximatively proportional to the number of scenarios, while the parallel time seems independent of the number of scenarios. Thus, we observe that the serial time of our approach in comparison with Minos increases slower with the problem size. Consequently, for large problems with reasonable precision requirements, our method appears faster than Minos even in a serial computer.

Key words: Large-Scale Convex Programming, Saddle Points, Parallel Computing, Stochastic Optimization.

Large-Scale Convex Optimization via Saddle Point Computation

Markku Kallio
Charles H. Rosa

1. Introduction

The objective of this article is to develop a parallel algorithm for solving large scale convex optimization problems specified as follows. Let $x = (x_j) \in R^n$ and let $f(x)$ be a convex and differentiable function. Let $g_i(x)$, for $i \in L$, be convex and differentiable functions and $g_i(x)$, for $i \in E$, be affine functions, for finite index sets L and E. Denote $g_L(x) = (g_i(x))_{i \in L}$, $g_E(x) = (g_i(x))_{i \in E}$ and $g(x) = (g_i(x)) \in R^m$ with $m = |L| + |E|$. Let vectors $l \leq u \in R^n$ define simple bounds on x. Define $X = \{x \in R^n \mid l \leq x \leq u\}$, which may not be bounded. Assume that the gradients of $f(x)$ and $g_i(x)$ are bounded on bounded subsets of X. Consider the following convex programming problem:

$$\min_{x \in X} f(x) \tag{1.1}$$

$$g_L(x) \leq 0 \tag{1.2}$$

$$g_E(x) = 0 \tag{1.3}$$

Let $y = (y_i) \in R^m$ be the dual vector associated with the constraints (1.2) and (1.3), and let $Y = \{y \in R^m \mid y_i \geq 0, \quad \forall\, i \in L\}$ account for the sign constraints of the dual vector y. We define the standard Lagrangian $L(x,y)$ as

$$L(x,y) = f(x) + y^T g(x) \tag{1.4}$$

and the Lagrangian dual problem of (1.1)-(1.3) as

$$\max_{y \in Y} \inf_{x \in X} L(x,y). \tag{1.5}$$

A point $(\hat{x}, \hat{y}) \in X \times Y$ is a saddle point of L over $X \times Y$ if

$$L(\hat{x}, y) \leq L(\hat{x}, \hat{y}) \leq L(x, \hat{y}) \quad \forall (x,y) \in X \times Y. \tag{1.6}$$

It is well known that if $(\hat{x}, \hat{y})$ is such a saddle point, then $\hat{x}$ and $\hat{y}$ are optimal solutions for (1.1)-(1.3) and (1.5); see e.g. Rockafellar [10]. Furthermore, under suitable constraint qualifications, such optimal solutions and saddle points are equivalent.

Recently Kallio and Ruszczyński [5] proposed for linear programming an algorithm, which is interpreted as a procedure for finding the saddle point for the standard Lagrangian. The key of this iterative method is to calculate the directions based on gradients of the Lagrangian at perturbed points. This procedure was extended in [6] to a class of saddle point problems, where the function $L(x, y)$ is convex in x and concave in y, and finite in a closed and convex set $X \times Y \subset R^n \times R^m$. Obviously, the saddle point problem (1.6) satisfies such requirements. Therefore in principle, the perturbation method of [6] applies to convex optimization. In this article we shall develop suitable methods for perturbation and scaling in order to make such a method work efficiently in practice. Following Murtagh and Saunders [9], we think of the constraints (1.2) and (1.3) as a large-scale and sparse system, which does not necessarily possess other structural properties.

In Section 2, we shall specialize the perturbation method of [6] to the saddle point problem (1.6). In particular, we propose a simple and computationally efficient procedure for the perturbation. In Section 3, our computer implementation together with proposed scaling procedures is presented. This implementation is embedded into GAMS [1]. Finally, preliminary numerical experience is reported in Section 4 on two sets of nonlinear stochastic optimization problems: multi-stage optimization models of the US energy sector, and multi-currency bond portfolio optimization problems.

2. The method

The general idea of the saddle point algorithm [6] is, in each iteration, to adjust primal and dual variables in the directions of the gradients. The adjusted points obtained are subsequently projected onto feasibility sets X and Y. These gradients are evaluated at perturbed points rather than at the solution at hand at the beginning of the iteration. For the step size, a simple rule is given to guarantee convergence. We shall first discuss the perturbation and propose mappings, which result in an efficient implementation. Thereafter, the algorithmic steps shall be stated. Finally, convergence of the convex optimization procedure is discussed.

The general aim of perturbation is to define mappings $\xi = \xi(x, y)$ and $\eta = \eta(x, y)$ so that the gap

$$E(x, y) = L(x, \eta) - L(\xi, y) \tag{2.1}$$

is positive. Unless (x, y) is a saddle point, such a mapping always exists. We decompose $E(x, y)$ into $E_x(x, y) + E_y(x, y)$ with

$$E_x(x, y) = L(x, y) - L(\xi, y) \tag{2.2}$$

$$E_y(x, y) = L(x, \eta) - L(x, y) \tag{2.3}$$

As discussed in [6], there are many ways of specifying such perturbations. If the feasible sets X and Y are compactified, we may define

$$\begin{aligned} \xi(x, y) &\in \arg\min_{x \in X} L(x, y) \\ \eta(x, y) &\in \arg\max_{y \in Y} L(x, y). \end{aligned}$$

To relax the boundedness requirements, some regularizing terms may be added in the above optimization problems. In any event, due to optimization, such a perturbation can be expensive.

A more practical approach is obtained via gradient perturbation. Denote the gradients of $L(x,y)$ by $e_x = \nabla_x L(x,y)$ and $e_y = \nabla_y L(x,y)$, and define the perturbations via gradient steps:

$$\xi(x,y) = [x - \alpha e_x]_X \tag{2.4}$$

$$\eta(x,y) = [y + \alpha e_y]_Y\,, \tag{2.5}$$

where $\alpha \geq 0$, and $[\ ]_X$ and $[\ ]_Y$ refer to orthogonal projections onto sets X and Y. We might then adopt line search techniques of nonlinear optimization to find a suitable value α in (2.4)-(2.5). We begin the perturbation with some positive trial step size and apply some suitable test to determine whether the step is small enough to yield an acceptable value for the gap function $E(x,y)$. If such test fails, the step size is reduced until the test is passed. Noting that the gap $E(x,y)$ as a function of α may be neither differentiable nor concave, a suitable test may be expensive to carry out. For that reason a further refinement is introduced as follows: instead of taking the perturbation direction e_x and e_y as gradients of $L(x,y)$, let $\bar{\alpha} > 0$ be a constant and define

$$e_x = -([x - \bar{\alpha}\nabla_x L(x,y)]_X - x)/\bar{\alpha} \tag{2.6}$$

$$e_y = \ \ ([y + \bar{\alpha}\nabla_y L(x,y)]_Y - y)/\bar{\alpha} \tag{2.7}$$

Consequently, if the perturbation is defined by (2.4) and (2.5), then the gap $E(x,y)$ becomes a concave and differentiable function for α in $(0,\bar{\alpha})$. In this interval, the Goldstein test (see e.g. [3], p. 27) will be applicable to determine a suitable stepsize α for perturbation; see Lemma1 below. If the test fails, a simple and practical rule is to reduce the step size α by a factor $1-\theta$, with $0 < \theta < 1$.

In general, the perturbation steps α in (2.4)- (2.5) may be specified individually for each primal and dual variable. Thereby we may exploit the observation that $L(x,y)$ is linear in y, and often in practice, linear in most of the primal variables x_j as well. Besides, these step sizes may vary from one iteration to another in the saddle point algorithm. In the following, however, we shall restrict the discussion to the case where a single iteration dependent stepsize α_x is applied to all primal variables and another α_y is applied to the dual variables. In particular, we choose a constant step size $\alpha_y = \bar{\alpha}$ to be applied to all dual variables. For primal variables, we search for a suitable step size $\alpha_x \in [0,\bar{\alpha}]$ employing the Goldstein test to $E_x(x,y)$ with a parameter $\omega \in (0,1)$. In a given iteration of the saddle point algorithm, the step size α_x found in the preceding iteration is first upgraded by a factor $1+\theta$ and then projected in the interval $[\check{\alpha},\bar{\alpha}]$, where $\check{\alpha} > 0$ is a minimum initial value applied through all iterations. The resulting value α_x is employed to begin the search for a step size in the current iteration. Following this line of thought, the perturbation mapping for (x,y) is formalized as follows:

Perturbation Mapping

Begin. Enter the perturbation routine with primal step size $\alpha_x > 0$ and dual step size $\alpha_y = \bar{\alpha}$. Replace α_x by $\min[\ \bar{\alpha},\ \max(\ \check{\alpha},\ (1+\theta)\alpha_x\ )]$. Find e_x and e_y according to (2.6)-(2.7).

Trial. Determine $\xi(x,y)$ by (2.4) with step size $\alpha = \alpha_x$, and $\eta(x,y)$ by (2.5) with step size $\alpha = \bar{\alpha}$.

Completion. Let $\psi(\alpha) = \psi(\alpha, x, y)$ denote the gap $E_x(x, y)$ as a function of α, and let $\psi' = \psi'(x, y)$ be the right derivative of $\psi(\alpha)$ at $\alpha = 0$. If $\psi' = 0$, set $\xi(x, y) = x$. If $\psi' = 0$ or if $\psi' > 0$ and $\psi(\alpha_x) \geq \omega\alpha_x\psi'$ (the Goldstein test), then the perturbation is completed. Otherwise, replace α_x by $(1 - \theta)\alpha_x$ and return to Trial. □

For convergence, the following result shall be employed:

Lemma 1. *The Perturbation Mapping satisfies the following conditions:*

1. *The vectors $\xi(x, y)$ and $\eta(x, y)$ are bounded on bounded subsets of $X \times Y$.*
2. *$E(x, y) \geq 0 \quad \forall(x, y) \in X \times Y$.*
3. *For every $(x, y) \in X \times Y$, if there is a sequence $(x^k, y^k) \to (x, y)$ such that $E(x^k, y^k) \to 0$, then (x, y) is a saddle point of L on $X \times Y$.*

Proof: Condition 1 follows directly from boundedness of perturbation steps sizes and from our assumption that the gradients $\nabla f(x)$ and $\nabla g_i(x)$, for all i, are bounded over bounded subsets of $X \times Y$.

Observing the Goldstein test for primal perturbation, we have $E_x(x, y) = \psi(\alpha_x, x, y) \geq \omega\alpha_x\psi'(x, y)$, with $\psi'(x, y) = e_x^T\nabla_x L(x, y) \geq 0$. Thus, $E_x(x, y) \geq 0$, and $E_y(x, y) = ([y + \bar{\alpha}g(x)]_Y - y)g(x) \geq 0$, so that $E(x, y) = E_x(x, y) + E_y(x, y) \geq 0$, and Condition 2 follows.

For Condition 3, let us assume the contrary: assume that $(\bar{x}, \bar{y})$ is not a saddle point and $(x^k, y^k) \to (\bar{x}, \bar{y})$ such that $E(x^k, y^k) \to 0$. For $\epsilon > 0$, define closed neighborhoods $B_\epsilon = \{(x, y) | \, \|(x - \bar{x}, y - \bar{y})\| \leq \epsilon\}$.

If $E_y(\bar{x}, \bar{y}) > 0$, then by continuity of $E_y(x, y)$, $\inf\{E_y(x, y) | (x, y) \in B_\epsilon\} > 0$, for small enough $\epsilon > 0$. As $E(x, y) \geq E_y(x, y)$, we have a contradiction with $E(x^k, y^k) \to 0$.

If $E_y(\bar{x}, \bar{y}) = 0$, we shall show, that in a neighborhood B_ϵ, $\psi'(x, y)$ as well as the step size α_x are bounded below by a positive numbers. This together with the Goldstein condition implies that there is $\nu > 0$ such that $E_x(x, y) \geq \omega\alpha_x\psi'(x, y) > \nu$, $\forall(x, y) \in B_\epsilon$; a contradiction with $E(x^k, y^k) \to 0$.

As $(\bar{x}, \bar{y})$ is not a saddle point, $E_y(\bar{x}, \bar{y}) = 0$ implies $\psi'(\bar{x}, \bar{y}) > 0$. In (2.6), the mapping $e_x : X \times Y \to R^n$ is continuous in $X \times Y$ and $\nabla f(x)$ is continuous in X. Therefore, $\psi'(x, y) = e_x^T\nabla_x L(x, y)$ is continuous in $X \times Y$. Consequently, for small enough $\epsilon > 0$, $\inf\{\psi'(x, y) | (x, y) \in B_\epsilon\} > 0$.

Finally, to investigate possible values of α_x, define the function $\varphi(\alpha, x, y) = \psi(\alpha, x, y) - \omega\alpha\psi'(x, y)$ as the excess in the Goldstein test. Define α^* so that $\varphi(\alpha^*, \bar{x}, \bar{y}) > 0$. Then, by continuity, $\varphi(\alpha^*, x, y) > 0$, $\forall(x, y) \in B_\epsilon$, with small enough $\epsilon > 0$. As $\varphi(\alpha, x, y)$ is concave in α, for $\alpha \in [0, \check{\alpha}]$, $\varphi(\alpha, x, y) > 0$, $\forall\alpha \in (0, \alpha^*)$, $\forall(x, y) \in B_\epsilon$, with $\epsilon > 0$ small enough. In this region, the Perturbation Mapping would not reduce the step size α_x any further. Hence, for the completion step size we have $\alpha_x \geq (1 - \theta)\min[\alpha^*, \check{\alpha}] > 0$, for all $(x, y) \in B_\epsilon$, with $\epsilon > 0$ small enough. □

The method for finding a saddle point is now stated as follows:

Saddle Point Algorithm

Initialization. Choose $x^0 \in X$, $y^0 \in Y$. Choose parameters $\gamma \in (0,2)$, $\bar{\alpha} \geq \breve{\alpha} > 0$ and ω, $\theta \in (0,1)$. Set $k = 0$.

Perturbation. Find perturbed points $\eta^k = \eta(x^k, y^k)$ and $\xi^k = \xi(x^k, y^k)$ employing the Perturbation Mapping.

Stopping test. Determine the gap $E_k = L(x^k, \eta^k) - L(\xi^k, y^k)$. If $E_k = 0$, then stop.

Update. Find gradients $L_x(x^k, \eta^k)$ and $L_y(\xi^k, y^k)$ and define

$$\begin{aligned} d_x^k &= \left[-L_x(x^k, \eta^k)\right]_{X_k} \\ d_y^k &= \left[L_y(\xi^k, y^k)\right]_{Y_k} \end{aligned}$$

where X_k and Y_k are cones of feasible directions for x and y, respectively, determined by binding simple bounds at (x^k, y^k), and $[\ ]_{X_k}$ and $[\ ]_{Y_k}$ refer to orthogonal projections on these cones.

Define

$$x^{k+1} = \left[x^k + \tau_k d_x^k\right]_X \tag{2.8}$$

$$y^{k+1} = \left[y^k + \tau_k d_y^k\right]_Y, \tag{2.9}$$

where the stepsize τ_k is given by

$$\tau_k = \frac{\gamma E_k}{\|d_x^k\|^2 + \|d_y^k\|^2}, \tag{2.10}$$

Increase k by one and go to Perturbation. □

As a consequence of Lemma 1 and the saddle point theorem of convex programming (Rockafellar [10]), a proof for the following convergence result may be adopted from [6]:

Theorem 1. *Assume that a saddle point of L on $X \times Y$ exists, or that an optimal solution for (1.1)-(1.3) exists, and that $0 \in \operatorname{int} g_E(X)$, and that $g_L(x^*) < 0$ and $g_E(x^*) = 0$, for some $x^* \in X$. Then the Saddle Point Algorithm generates a sequence $\left\{(x^k, y^k)\right\}_{k=0}^{\infty}$ convergent to an optimal solution of (1.1)-(1.5).*

3. Implementation

An experimental computer code has been developed on the basis of Section 2. We shall call our implementation *Convex*. The initial experiments indicate that scaling is crucial for an efficient implementation. We shall shortly discuss the dynamic scaling procedure in *Convex*. Data storage and computational steps shall be outlined thereafter. Finally, we discuss the linkage of *Convex* with GAMS [1]. At this point we wish to stress, that all steps in *Convex* have a great potential for parallelization: all primal and dual variables can be processed in parallel. Besides, sparsity can be exploited in communication as well. Therefore the approach is well suited for massively parallel computing. However, our preliminary tests with *Convex* shall be executed in a serial computer.

Scaling

The procedure employed in the linear programming code by Kallio and Salo [7] shall be adjusted for *Convex*. Denote $c = (c_j) = \nabla f(x) \in R^n$ and denote by $A = (a_{ij}) \in R^{m\times n}$ the Jacobian of $g(x)$, so that $a_{ij} = \partial g_i(x)/\partial x_j$. In each iteration, first, auxiliary *reference quantity* and *value* vectors $\epsilon' = (\epsilon'_i) \in R^m$ and $\delta' = (\delta'_j) \in R^n$ are defined so that

$$\begin{aligned} \epsilon'_i &= \sum_j |x_j a_{ij}| \\ \delta'_j &= \sum_i |y_i a_{ij}| + |c_j|, \end{aligned}$$

where x_j and y_i denote primal and dual solutions at the beginning of the iteration. Denote by δ^- and ϵ^- these reference vectors at the beginning of the iteration. For the first iteration, these initial values are set to σ, which also represents an exogenous lower bound for the reference quantities and values during the iterations. The updated reference vectors δ and ϵ for the current iteration are computed as

$$\begin{aligned} \delta_j &= \max(\sigma, \beta\delta'_j + (1-\beta)\delta^-_j), \\ \epsilon_j &= \max(\sigma, \beta\epsilon'_j + (1-\beta)\epsilon^-_j), \end{aligned}$$

where β is an exponential smoothing parameter. Its purpose is to prevent erratic behavior of reference vectors over the iterations.

For scaling the primal and dual variables, we define diagonal matrices $G \in R^{n\times n}$ and $D \in R^{m\times m}$ with positive diagonal elements G_j and D_i, respectively. Primal variables x_j shall then be scaled by $(G_j)^{-1/2}$ and the dual variables y_i by $(D_i)^{-1/2}$; i.e. G_j and D_i are squared column and row scalers, respectively.

Elements G_j and D_i are given by

$$G_j = \rho H_i \left\{ \frac{\epsilon_i}{|a_{ij}|\delta_j} \right\} \tag{3.1}$$

$$D_i = \rho H_j \left\{ \frac{\delta_j}{|a_{ij}|\epsilon_i} \right\}, \tag{3.2}$$

where $\rho > 0$ is a constant and operators H_i and H_j refer to harmonic means over i and j, for a_{ij} nonzero.

Also the factors G_j and D_i will be updated in the course of the iterations. We shall first apply (3.1) and (3.2) for obtaining auxiliary factors. The factors employed for scaling in *Convex* are then obtained via exponential smoothing over iterations. Again, we employ the weight β for the auxiliary factors and $1-\beta$ for the factors employed in the preceding iteration. For the first iteration such initial values are set equal to a constant $\kappa > 0$.

The Data Structure

As in Minos [9], the (potentially) nonzero data for the gradient c and the Jacobian A is stored columnwise accounting for sparsity. For the purpose of dual updates, the locations of nonzero elements of c and A are stored row-wise as well. Bounds l and u are stored as dense vectors.

Iterative Steps

We shall now outline the computations carried out by *Convex.* The algebraic steps are taken in scaled primal and dual spaces, while actual computations are carried out in the unscaled spaces. Thus problem data shall not be scaled, but appropriate factors are employed in the steps of the algorithm to account for scaling. The computational steps may now be specified as follows:

1. To begin the first iteration, set all primal variables x_j to a constant value x^0 and project onto the simple bounds. Set all dual variables y_i to zero. Assign a constant value $\kappa > 0$ to all factors G_j and D_i. Set all reference quantities ϵ_i and values δ_j to a constant σ. Choose the update step size parameter $\gamma \in (0,2)$, the perturbation step size parameters $\bar{\alpha} \geq \breve{\alpha} > 0$ and $\omega,\ \theta \in (0,1)$, and a stopping tolerance $\phi > 0$.

2. Evaluate the objective function $f(x)$, the gradient $c = \nabla f(x)$, the constraint function $g(x)$ and the Jacobian $A = \partial g(x)/\partial x$.

3. Find perturbation $\eta_i(x,y)$ and update factors D_i, for all i.

4. Find a trial perturbation $\xi_j(x,y)$ and update factors G_j, for all j.

5. Determine the gap $E(x,y)$ and carry out the Goldstein test for the primal perturbation stepsize. If the test fails, reduce the perturbation step and return to Step 4.

6. Evaluate $g_i(\xi)$; i.e. the gradient component of $L(\xi,y)$, for each i, and determine the direction d_y.

7. Evaluate the gradient component $\partial L(x,\eta)/\partial x_j$, for each j, and determine the direction d_x.

8. Determine the step size according to (2.10).

9. Update dual variables according to (2.9).

10. Update primal variables according to (2.8).

11. If $E(x,y) > \phi|f(x)|$, return to Step 2; otherwise stop.

Linkage with GAMS

GAMS (General Algebraic Modeling System) is a modeling language that enables end users to describe their convex optimization problems in a relatively clear and logical mathematical programming format, and then solve using a variety of industry standard solvers. GAMS becomes useful to optimization code developers by virtue of it being possible to attach any solver to GAMS and effectively use GAMS as a problem generator and function evaluater. This frees the developers to concern themselves solely with the solver itself. This is what we did in order to easily assess our methods applicability to existing GAMS models, in particular the Global 2100 based multi-scenario energy/economic model of the US.

We affected the linkage via a library of subroutines called cplib [2]. This library was originally designed to facilitate the connection of mixed nonlinear complementarity and variational inequality solvers to GAMS. With some minor modifications to the library routines, though, the *Convex* optimization solver became compatible. The most important features of the library are the routines enabling the retrieval of function, gradient and Jacobian information (CPFUNF(.), CPSPRJ(.)). These features are used during each iteration of the algorithm, calculating the required gradients of the lagrangian function that provide the directions to the perturbed points and the gradients at the perturbed points themselves. The library routines also automatically handle such mundane but important tasks as the allocation of memory for the solvers core space, the passing back to GAMS of the state of the solution at solver termination, and the solution itself. A complete description of the library's capabilities can be obtained from [2].

A linkage like just described greatly broadens the set of test problems available for analysis. Of course, we have also implemented a version of *Convex* that runs in stand-alone mode, obtaining Jacobian and function evaluation information from separate subroutines, as in Minos.

4. Computational Tests

Convex was tested on two classes of problems. The first one arises from the Global 2100 model by Manne and Richels [8]. Based on this model, Rosa [11] has developed a set of multi-stage stochastic optimization models for the US energy sector. The objective is to maximize the expected present value of a logarithmic utility function. The constraints include nonlinearities in production functions. In Table 1, problems US1,...,US16E refer to these models. Names ending with a letter E refer to formulations with explicit nonanticipativity constraints (see [11]), while those ending with the letter C refer to condensed problems with the usual block-angular structure. Thereby problem pairs US4C-US4E, US8C-US84E and US16C-US16E are equivalent formulations. The numbers in names indicate the number of scenarios in the problem. Thus US1 is in fact a deterministic dynamic problem. The second set of test problems consists of two-stage optimization of multi-currency bond portfolios by Huoponen [4]. The second stage problems themselves are stochastic optimization problems with ten scenarios. The objective is to maximize

	Total			Nonlinear		
Problem	Rows	Columns	Nonzeros	Rows	Columns	Nonzeros
US1	398	595	1846	23	110	99
US4C	1589	2023	7376	79	390	391
US4E	2201	2449	8605	89	440	396
US8C	3177	3655	14744	151	750	775
US8E	4809	4897	18025	177	880	792
US16C	6353	6919	29480	295	1470	1543
US16E	9617	9793	36049	353	1760	1584
B100	102	909	2709	1	900	900
B500	502	4509	13509	1	4500	4500
B1000	1002	9009	27009	1	9000	9000
B5000	5002	45009	135009	1	45000	45000
B10000	10002	90009	270009	1	90000	90000

Table 1: The number of rows, columns, non-zeros and non-linearities in the test problems.

the expected value of a negative exponential utility function of the return on investment. The constraints are linear budget constraints, including one for each first stage scenario. We refer to these problems by B100,...,B10000, where the numbers refer to the number of first stage scenarios. Dimensions of all test problems are given in Table 1.

All solution times reported below refer to cpu seconds on a SPARC 10 workstation operating under Solaris 2.3. Time for data input and output is excluded. The US energy models were written in GAMS. Therefore we use GAMS/Minos [1] for comparisons with *Convex*. Both runs employ routines provided by GAMS for function and gradient evaluations. The bond portfolio problems were solved with Minos 5.3 for such comparisons. The same subroutines were used for function and gradient evaluations both in Minos 5.3 and in *Convex*. Problem B10000 appeared too big to fit in our computer with Minos 5.3.

The solution times for Minos are obtained using default values for specs parameters with the following exception: For the portfolio problems, the upper limit on the number of superbasic variables allowed had to be increased to 1000. For *Convex*, initial values for primal variables x_j are set to one, for the energy models, and to zero, for the portfolio models. All dual variables y_i are equal to zero initially. The stepsize parameter $\gamma = 1.8$, the scaling parameter $\rho = 0.5$, the exponential smoothing weight $\beta = 0.5$, the initial reference quantity and value $\sigma = 0.01$, the initial value of scaling factors G_j and D_i is $\kappa = 0.1$. Scaling parameters were updated in the first 500 iterations and kept constant thereafter. For the perturbation step size parameters, the following values are used: $\bar{\alpha} = 1$, $\check{\alpha} = 10^{-6}$, $\theta = 0.5$, $\omega = 0.05$. For the termination parameter ϕ we use $\phi = 10^{-4}$ for the energy models, and $\phi = 10^{-5}$ for the portfolio models.

For *Convex*, runs were performed to obtain serial run times. Based on these serial times, we also calculated the amount of time required per unit of problem size where a problems size is defined to be $m + n$. We did this to illustrate the fact that the unit computational effort remains relatively constant as the problem size grows larger. This has major implications for the algorithms behavior when implemented in parallel, as the natural technique for such an implementation would be to assign a processor to each

	Minos	Convex			
	Solution	Serial	Unit		Relative
Problem	Time	Time	Effort	Iterations	Error
US1	10	12	0.012	325	5.E-4
US4C	212	54	0.015	439	1.E-3
US4E	214	54	0.012	333	3.E-4
US8C	1375	142	0.020	592	2.E-4
US8E	1211	105	0.011	292	3.E-4
US16C	6923	624	0.047	1317	3.E-3
US16E	7556	231	0.012	312	3.E-4
B100	12	92	0.091	2917	2.E-3
B500	325	518	0.103	2662	3.E-3
B1000	1637	994	0.099	2530	3.E-3
B5000	88061	4682	0.094	2432	2.E-3
B10000	-	9633	0.096	2462	2.E-3

Table 2: Solution time (sec) of GAMS/Minos, for the problems US1,...,US16E, and of Minos 5.3, for problems B100,...,B5000. Serial time and unit effort (sec), number of iterations and relative error in the optimal objective function using *Convex.*

column and row. Table 2 shows the serial run time for Minos and *Convex*, as well as the unit effort, iteration count and relative error in the optimal objective function value for *Convex.*

The general observation in these results is that the larger the problem, the more efficient *Convex* is relative to Minos. In a serial computer, for small problems, *Convex* is slower, but for the larger problems an adequate precision is found by *Convex* faster than by Minos. This conclusion holds even if Minos would be terminated at the precision achieved by *Convex*: the run time for Minos would then be reduced, but only by about one third. This may be explained as follows. Consider one of the problem sets and let s be the number of scenarios in a problem of this set. For Minos, the number of iterations increases proportionally with s and the work per iteration increases even faster. Thus the run time for Minos increases faster than s^2. For *Convex*, the number of iterations appears roughly independent of s (with some exception) while the work per iteration increases proportionally with s. Consequently, the serial run time appears proportional to s, and the theoretical run time in a massively parallel computer is independent of s. In closing, we note that even in calculating the correct perturbation mapping, an expensive part of algorithm in terms of computational effort, the backward step from the completion phase to the trial phase, which could conceivably cycle many times, usually only occurs once in two or more iterations in the problems we investigated.

5. Conclusions

A recent approach for saddle point computation [6] has been specialized to solve large-scale convex optimization problems with differentiable objective and constraint functions.

This iterative method proceeds in directions determined by gradients of the standard Lagrangian. Gradient evaluation takes place at perturbed points. A central topic in this article is to propose a perturbation procedure, which yields an efficient implementation in practical applications. In order to ensure efficiency, a scaling procedure was devised as well. The resulting method suits well to massively parallel computing.

An experimental code embedded into GAMS was tested in a serial computer on two sets of nonlinear problems: multi-stage stochastic optimization of the US energy economy, and multi-currency bond portfolio problems. These preliminary tests indicate that, for large problems with reasonable precision requirements, our method is faster than Minos. Thus the method can be very efficient for large problems, even in serial computing , but especially in parallel computing environments, where we can expect speedups of many orders of magnitude.

Acknowledgements

The authors wish to thank Andrzej P. Wierzbicki for fruitful discussions concerning the perturbation scheme, and in particular for referring us to the Goldstein test for line search. Constructive and encouraging comments by Yuri Ermoliev and Michael Saunders are gratefully acknowledged as well.

References

[1] A. Brooke, D. Kendrick and A. Meeraus, *GAMS: User's Guide, Release 2.25*, The Scientific Press, 1992.

[2] S.P. Dirkse, M.C. Ferris, P.V. Preckel, T. Rutherford, "The GAMS Callable Program Library for Variational and Complementary Solvers," Mathematical Programming Technical Report 94-07, Computer Sciences Department, University of Wisconsin, 1994.

[3] R. Fletcher, *Practical Methods of Optimization*, John Wiley & Sons, 1987.

[4] T. Huoponen, "Stochastic Optimization of a Multi-Currency Bond Portfolio," Working Paper WP-94-98, IIASA, Laxenburg, 1994.

[5] M. Kallio and A. Ruszczyński, "Parallel Solution of Linear Programs via Nash Equilibria," Working Paper WP-94-15, IIASA, Laxenburg, 1994.

[6] M. Kallio and A. Ruszczyński, "Perturbation Methods for Saddle Point Computation," Working Paper WP-94-38, IIASA, Laxenburg, 1994.

[7] M. Kallio and S. Salo, "Tatonnement Procedures for Linearly Constrained Convex Optimization," *Management Science* 40 pp 788-797 (1994).

[8] A.S. Manne and R.G. Richels, *Buying Greenhouse Insurance: The Economic Costs of CO_2 Emission Limits*, The MIT Press, 1992.

[9] B.A. Murtagh and M.A. Saunders, "Minos 5.1 User's Guide," Technical Report SOL 83-20R, Stanford University, 1987.

[10] R.T. Rockafellar, *Convex Analysis*, Princeton University Press, 1970.

[11] C.H. Rosa, "Pathways of Economic Development in an Uncertain Environment," Working Paper WP-94-41, IIASA, Laxenburg, 1994.