

# An interior point method and Sherman–Morrison formula for solving large scale convex quadratic problems with diagonal Hessians

Nadezda Sukhorukova<sup>1</sup>

January 25, 2015

## Abstract

We develop an approach for solving large scale convex quadratic problems with quadratic matrices subject to linear equalities and box-constraints. These problems appear in real-life applications. At first glance, this is a simple convex optimisation problem. However, the size of this problem ( $10^9$  variables and  $10^6$  constraints for some applications) makes it very challenging to apply traditional convex optimisation techniques. Therefore, one needs to develop a specific algorithm for solving such kind of problems. We apply a combination of the Interior Point method and Sherman–Morrison formula to solve this problem. We test our approach on smaller size datasets (1000 variables and 100 constraints). Our numerical experiments show that this combination is

---

<http://journal.austms.org.au/ojs/index.php/ANZIAMJ/article/view/7574> gives this article, © Austral. Mathematical Soc. 2015. Published January 25, 2015. ISSN 1446-8735. (Print two pages per sheet of paper.) Copies of this article must not be made otherwise available on the internet; instead link directly to this URL for this article.

efficient, fast and computationally stable. This approach is suitable for large scale convex quadratic optimisation problems.

*Keywords:* large scale convex quadratic problems, interior point methods, Sherman–Morrison formula, social accounting matrices

# Contents

<b>1</b>	<b>Introduction</b>	<b>E3</b>
<b>2</b>	<b>Interior Point Method</b>	<b>E6</b>
2.1	Equality constrained subproblem . . . . .	E6
2.2	A barrier method for inequality constrained optimisation . . . . .	E8
2.3	Computational concerns . . . . .	E10
<b>3</b>	<b>Matrix inverse: Cholesky decomposition and Sherman–Morrison formula</b>	<b>E10</b>
3.1	Inverse through Cholesky decomposition . . . . .	E10
3.2	Sherman–Morrison formula . . . . .	E11
<b>4</b>	<b>Sherman–Morrison formula in IPM</b>	<b>E12</b>
4.1	IPM implementation using Sherman–Morrison formula . . . . .	E12
4.2	Matrix update implementation using Sherman–Morrison formula . . . . .	E14
4.3	Advantages of solving linear systems through matrix inverse in SAMRPR . . . . .	E15
<b>5</b>	<b>Numerical experiments</b>	<b>E16</b>
<b>6</b>	<b>Conclusions and further research directions</b>	<b>E18</b>
	<b>References</b>	<b>E19</b>

# 1 Introduction

Our goal is to apply the Interior Point method to a large scale convex quadratic problem with linear and box-constraints. This problem appears in many real-life applications: social accounting matrix reconciliation problem (SAMRP) [16], geodesic problems [6], signal processing [12], and many others. The original SAMRP can be formulated as the minimisation of a convex quadratic function subject to linear equality and box constraints:

$$\min \mathbf{x}^T \mathbf{M} \mathbf{x}, \quad \text{subject to} \quad \mathbf{A} \mathbf{x} = \mathbf{b}, \quad \mathbf{x}_l \leq \mathbf{x} \leq \mathbf{x}_u, \quad (1)$$

where  $\mathbf{M}$  is a positive diagonal matrix,  $\mathbf{A}$  is a matrix of constraints (full rank),  $\mathbf{x}_l$  and  $\mathbf{x}_u$  are lower and upper bounds. The dimension of the optimisation problem in SAMRP is up to  $10^9$  and the number of linear constraints is up to  $10^6$ . At first glance, SAMRP is a simple convex optimisation problem. However, the size of this problem makes it very challenging to apply traditional convex optimisation techniques. When modelling the economy of a particular region, the size of the problem is significantly lower, but still around  $10^6$  variables and  $10^5$  constraints (SAMRP regional or SAMRPR). Usually a number of closely related SAMRPs for consecutive time periods need to be solved.

The constraint matrix  $\mathbf{A}$  in (1) represents the relationships between different industry sectors. Part of this matrix (so called balancing constraints, which ensure that the net monetary input into each industry sector is the same as its net monetary output) is the same every period. Other constraints may change from one period to another (aggregation and disaggregation constraints, which ensure that certain partnership agreements are met), but this change is not very significant, since it takes much time and resources to change the established relationships. Also, in many cases two industries, which are not normally related to each other, will stay unrelated in the next time period. Therefore, only a few elements of the constraint matrix should be updated to obtain the constraint matrix for the next period. Another important issue is that the balancing constraints are much more dense than the aggregation/disaggregation constraints.

*Example 1.* Consider (1) with  $10^8$  variables and  $10^5$  constraints. This means that the original social accounting matrix contained  $10^4 = \sqrt{10^8}$  industry sectors (the variables in SAMRP are all the entries in of a square matrix of size  $10^4$ , this matrix represents the exchange between the sectors). Then the number of balancing constraints is  $10^4$  and the rest ( $10^5 - 10^4$ ) are the aggregation/disaggregation constraints. Therefore, the balancing constraints constitute around 10% of all the constraints (same for every period) and 90% are aggregation/disaggregation constraints (differ from one period to another, but not significantly).

The rest of this Introduction briefly discusses several main approaches to solve this problem:

- Gaussian Elimination method (GEM);
- Conjugate Gradient method (CGM);
- Augmented Lagrangian method (ALM); and
- Interior Point method (IPM).

GEM and CGM can be used to solve a relaxation of (1) without box-constraints. Some additional analysis is necessary to ensure that the box-constraints also hold.

GEM is much older than CGM and IPM. The main advantage of this group of methods is that when the solution is obtained this would be the exact solution rather than its approximation (CGM and IPM). At the same time, if for some reason the procedure was not completed (for example, high computational time), then there would be no improvement (in contrast, CGM and ALM improve the objective function value at every iteration). Godzio and Grothey [8] reported that for some applications (when the structure of the constraint matrix is known to fit into certain patterns) they were able to solve a problem with over  $10^9$  variables using GEM. The program ran on 1280 processors.

CGM is a very efficient class of iterative methods for solving sparse systems

of linear equations with real symmetric positive definite matrices, originally proposed by Hestenes and Stiefel [10]. Its convergence is improved drastically by applying an appropriate preconditioner (Golub and Van Loan [7]). However, the problem of constructing this preconditioner is itself a very challenging problem.

ALMs are a certain class of algorithms for solving constrained optimisation problems, where a constrained optimisation problem is replaced by a series of unconstrained problems. These methods were first proposed by Hestens [9] and Powell [15]. Since the 1980s, IPM have had increasing attention. However, according to Nocedal and Write [14], ALM is also very efficient.

IPM (Barrier Methods) are a certain class of methods to solve linear and nonlinear convex optimisation problems. These algorithms have been inspired by Karmarkar's algorithm [11]. The original Karmarkar's algorithm was not computationally efficient outside of the class of linear problems, but its modern implementations are computationally efficient on an extremely wide class of convex problems (Nesterov and Nemirovskii [13]). One of the disadvantages of this group of methods is that one needs to solve large linear systems at each iteration. This can be done by CGM and GEM. We propose a modification of IPM which significantly simplifies this process. Our approach is based on the Sherman–Morrison formula [17], which enables one to calculate the inverse to a matrix update when the inverse of the original matrix is known (or can be computed inexpensively). In many cases the calculation of matrix inverse is not the best way to solve linear systems [5]. However, for some particular application (SAMRPR) this kind of approach is efficient (section 4). Dongarra et al. [4] demonstrated that in some practical problems it is necessary to calculate the inverse explicitly. Also, they calculated the inverse of dense matrices ( $10000 \times 10000$ ) using LAPACK [1].

Kim et al. [12] modified IPM, applying CGM with certain preconditioners to solve large scale least squares problems appearing in signal processing and statistics. They report that their method is able to solve large sparse problems with a million variables with high accuracy in a few tens of minutes

on a PC. These results inspired us to develop a modification of IPM, which would be able to solve (1) efficiently.

We work with the important special case

$$\min \mathbf{x}^T \mathbf{M} \mathbf{x}, \quad \text{subject to} \quad \mathbf{A} \mathbf{x} = \mathbf{b}, \quad \mathbf{x} \geq 0, \quad (2)$$

where the simple bounds of (1) are replaced by non-negativity constraints.

## 2 Interior Point Method

Karush–Kuhn–Tucker conditions (KKT) have been obtained for smooth optimisation problems. In most cases, the KKT conditions is a system of equalities and inequalities. In the case of linear programming problems and some other classes of convex problems (including SAMRPR) these conditions are necessary and sufficient optimality conditions. Therefore, solving SAMRPR (or SAMRP) and solving the corresponding KKT system are equivalent. Most IPM can be interpreted as methods for solving KKT systems.

### 2.1 Equality constrained subproblem

Consider the following convex optimisation problem with equality constraints, for example, a relaxation of the original problem (1) without box-constraints:

$$\min f(\mathbf{x}) \quad \text{subject to} \quad \mathbf{A} \mathbf{x} = \mathbf{b}, \quad (3)$$

where  $f(\mathbf{x}) = \mathbf{x}^T \mathbf{M} \mathbf{x}$ , and  $\mathbf{M}$  is a positive definite matrix. Then the corresponding KKT system is

$$\mathbf{A} \mathbf{x}^* = \mathbf{b}, \quad \nabla f(\mathbf{x}^*) + \mathbf{A}^T \mathbf{v}^* = 0, \quad (4)$$

where  $\mathbf{x}^*$  is an optimal solution for (3), and  $\mathbf{v}^*$  is the corresponding dual variable.

Let  $\mathbf{x}^* = \mathbf{x} + \Delta\mathbf{x}$ , where  $\Delta\mathbf{x}$  is the Newton step which has to be added to the initial strictly feasible  $\mathbf{x}$  (that is, feasible and the box-constraints are satisfied as strict inequality constraints) to solve (3). Then the above KKT can be reformulated as

$$\begin{cases} \mathbf{A}(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{b}, \\ \nabla f(\mathbf{x} + \Delta\mathbf{x}) + \mathbf{A}^T \mathbf{v}^* = 0; \end{cases} \iff \begin{cases} \mathbf{A}\Delta\mathbf{x} = 0, \\ \nabla f(\mathbf{x}) + \nabla^2 f(\mathbf{x})\Delta\mathbf{x} + \mathbf{A}^T \mathbf{v}^* = 0. \end{cases} \quad (5)$$

Therefore, the final system to obtain  $\Delta\mathbf{x}$  is

$$\begin{bmatrix} \nabla^2 f(\mathbf{x}) & \mathbf{A}^T \\ \mathbf{A} & 0 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \mathbf{v}^* \end{bmatrix} = \begin{bmatrix} -\nabla f(\mathbf{x}) \\ 0 \end{bmatrix}. \quad (6)$$

If the problem can be formulated as (relaxation without box-constraints)

$$\min f(\mathbf{x}) = \mathbf{x}^T \mathbf{M} \mathbf{x}, \quad \text{subject to} \quad \mathbf{A} \mathbf{x} = \mathbf{b}, \quad (7)$$

then it can be solved in one step by assigning  $\mathbf{x}^* = \mathbf{x} + \Delta\mathbf{x}$ , where  $\mathbf{x}^*$  is the optimal solution to (7),  $\mathbf{x}$  is a starting solution (has to be feasible), and  $\Delta\mathbf{x}$  is a Newton step obtained by solving

$$\begin{bmatrix} 2\mathbf{M} & \mathbf{A}^T \\ \mathbf{A} & 0 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \mathbf{v}^* \end{bmatrix} = \begin{bmatrix} -2\mathbf{M}\mathbf{x} \\ 0 \end{bmatrix},$$

where  $\mathbf{v}^*$  is a dual variable.

If the objective function  $f(\mathbf{x})$  is not quadratic, then more than one Newton step may be necessary. The corresponding algorithm is listed in Algorithm 1.

This algorithm is similar to Algorithm 11.1 from Boyd's book [3]. The next step is to introduce an algorithm which can also take into account the inequality constraints (box-constraints).

**Algorithm 1:** Pseudocode for Barrier Method's equality subproblem**Data:** Start with a feasible initial solution  $\mathbf{x}_0$ ,  $\varepsilon > 0$  is the accuracy.**1 repeat****2**     Compute the Newton step  $\Delta\mathbf{x}$  by solving

$$\begin{bmatrix} \nabla^2 f(\mathbf{x}) & \mathbf{A}^\top \\ \mathbf{A} & 0 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \mathbf{v}^* \end{bmatrix} = \begin{bmatrix} -\nabla f(\mathbf{x}) \\ 0 \end{bmatrix}, \quad (8)$$

where  $\nabla^2 f(\mathbf{x})$  is the Hessian,  $\nabla f(\mathbf{x})$  is the gradient. ;**3**     Compute the Newton decrement  $\lambda^2 = \Delta\mathbf{x}^\top \nabla^2 f(\mathbf{x}) \Delta\mathbf{x}$ . (The value  $\lambda^2/2$  gives a good estimate of the difference between the current value of the objective function and the optimal value.) ;**4**     **if**  $\lambda^2/2 > \varepsilon$  **then****5**         Choose a step size  $\alpha$  (line search). (Different types of line search may be used, for example, backtracking line search.) ;**6**         Update  $\mathbf{x} = \mathbf{x} + \alpha\Delta\mathbf{x}$ . ;**7**     **end****8 until**  $\lambda^2/2 \leq \varepsilon$ ;

## 2.2 A barrier method for inequality constrained optimisation

In the case of inequality constrained optimisation, the KKT conditions are more complex than they are in the case of equality constrained minimisation. In this case the Barrier Method is based on solving a sequence of unconstrained minimisation problems (or problems with linear equality constraints only, see section 2.1).

The problem is now formulated as

$$\min \frac{k}{\varepsilon} \mathbf{x}^\top \mathbf{M} \mathbf{x} + \varphi(\mathbf{x}), \quad \text{subject to} \quad \mathbf{A} \mathbf{x} = \mathbf{b}, \quad (9)$$

where  $\varepsilon$  is the accuracy,  $k$  is the number of functions which contribute to



---

**Algorithm 2:** Pseudocode for Barrier Method (inequality constraints)

---

**Data:** Start with a strictly feasible initial solution  $\mathbf{x}_0$ ,  $\mathbf{t} = \mathbf{t}_0 > 0$ ,  
 $\mu > 1$ ,  $\varepsilon > 0$ .

**1 repeat**

**2** (repeat until we reach an  $\varepsilon$ -suboptimal solution of the original problem.);

**3** Centering Step: Compute  $\mathbf{x}^*(\mathbf{t})$  as a solution of the problem

$$\min \mathbf{t}\mathbf{x}^T\mathbf{M}\mathbf{x} + \varphi(\mathbf{x}), \quad \text{subject to } \mathbf{A}\mathbf{x} = \mathbf{b}, \quad (10)$$

starting at  $\mathbf{x}$ . (This can be done through the Barrier Method (equality constraints): the new objective function is  $\mathbf{t}\mathbf{x}^T\mathbf{M}\mathbf{x} + \varphi(\mathbf{x})$ .);

**4** Update  $\mathbf{x} = \mathbf{x}^*(\mathbf{t})$ ,  $\mathbf{t} = \mu\mathbf{t}$ ;

**5 until**  $k/t < \varepsilon$ ;

---

box-constraints (one per box constraint if the upper bound is unlimited or two per box-constraint otherwise),  $\varphi(\mathbf{x})$  is a barrier function (we are concentrating on logarithmic barrier functions).

The function  $\varphi(\mathbf{x}) = -\sum_{i=1}^k \log(f_i(\mathbf{x}))$  is a logarithmic barrier function (LBF), where  $f_i(\mathbf{x}) = x_i$ ,  $i = 1, \dots, k$ , are the nonnegativity constraints. LBF grows without bounds if  $f_i(\mathbf{x}) \rightarrow 0$  (at the barrier of the feasible region) and therefore it ‘helps’ to keep the solutions feasible.

The final algorithm is listed in Algorithm 2.

For box-constraints with all variables to be nonnegative (that is,  $\mathbf{x} \geq 0$ )  $\nabla^2\varphi(\mathbf{x}) \in \mathbb{R}^{n \times n}$  is a diagonal matrix with the  $i$ th diagonal element equaling  $1/x_i^2$ , and the gradient  $\nabla\varphi(\mathbf{x}) = -(1/x_1, \dots, 1/x_n)^T \in \mathbb{R}^n$ .

Therefore, IPM enables one to reduce SAMRPR to solving a sequence of large linear system. One way to solve these systems is through CGM or GEM. Another way is to apply Sherman–Morrison formula (see section 3 for details).

## 2.3 Computational concerns

There are two main issues which should be studied before applying IPM.

- First of all, it is essential to find a strictly feasible initial solution. If this solution is not obvious, then one can find such a solution by solving

$$\min \frac{k}{\varepsilon}, \quad \text{subject to} \quad \mathbf{Ax} = \mathbf{b}, \quad -f_i(\mathbf{x}) \leq s, \quad i = 1, \dots, k. \quad (11)$$

In some cases, this stage of obtaining a strictly feasible solution is called Phase 1 and then the actual Barrier Method (inequality constraints) is called Phase 2.

- The second problem is how to solve the linear systems appearing in (8). This problem has to be solved several times at Step 1 of the Barrier Method (inequality constraints). A possible approach to solve this system is indicated in section 3.

# 3 Matrix inverse computation: Cholesky decomposition and Sherman–Morrison formula

## 3.1 Inverse through Cholesky decomposition

We apply a well-known two stage approach for calculating the inverse of a positive definite matrix [7].

- Calculate a decomposition of the original matrix into the product of a lower triangular matrix and its transpose  $\mathbf{B} = \mathbf{LL}^T$  (Cholesky decomposition).
- Calculate the inverse of  $\mathbf{L}$  and construct  $\mathbf{B}^{-1} = (\mathbf{L}^{-1})^T \mathbf{L}^{-1}$ .

Both stages of this procedure allow a certain level of parallelization, which is essential in the case of large dimensional problems.

### 3.2 Sherman–Morrison formula

The Sherman–Morrison [2, 17] formula enables one to calculate the inverse of a matrix

$$\tilde{W} = W + \mathbf{V}\mathbf{U}^T, \tag{12}$$

where  $W$  is an easily invertible matrix, and  $\mathbf{U}, \mathbf{V} \in \mathbb{R}^n$ : namely,

$$\tilde{W}^{-1} = W^{-1} - \frac{W^{-1}\mathbf{V}\mathbf{U}^T W^{-1}}{1 + \mathbf{U}^T W^{-1}\mathbf{V}}. \tag{13}$$

The above formulation first appeared in Bartlett’s paper [2]. One problem with applying this formula is that in some cases we cannot be sure that the denominator

$$1 + \mathbf{V}^T W^{-1}\mathbf{U} \neq 0.$$

Sherman and Morrison [17] originally worked with a matrix update, where only one element is to be updated at a time; for example, for updating  $w_{pq}$  (this element corresponds to the  $p$ th row and  $q$ th column of the original matrix  $W$ ) one should use  $\mathbf{U}$  and  $\mathbf{V}$ , such that all the elements in these vectors are zero, except  $u_p$  and  $v_q$ , namely,  $u_p v_q = \delta$ , where  $\delta$  is the update for the element  $w_{pq}$ .

Sherman and Morrison mentioned [17] that for this type of update, the equality  $1 + \mathbf{V}^T W^{-1}\mathbf{U} = 0$  means that  $\tilde{W}$  is singular. However, the formal proof was left for the reader to complete.


In more general cases, when more than one element is updated, the following theorem applies.

**Theorem 2.** *If both  $W$  and  $\tilde{W} = W + \mathbf{V}\mathbf{U}^T$  are invertible, then  $1 + \mathbf{U}^T W^{-1}\mathbf{V} \neq 0$ .*

**Proof:** Suppose that both  $W$  and  $\tilde{W} = W + \mathbf{V}\mathbf{U}^\top$  are invertible, but  $1 + \mathbf{U}^\top W^{-1}\mathbf{V} = 0$ . In this case  $\tilde{W}W^{-1}$  is non-singular and therefore  $\tilde{W}W^{-1}\mathbf{V} \neq \mathbf{0}$ , unless  $\mathbf{V} = \mathbf{0}$ , where  $\mathbf{0}$  means “zero-vector”, that is, all the coordinates are zeros.

The case when  $\mathbf{V} = \mathbf{0}$  is not interesting, since this means that there is no matrix update. Moreover, if  $\mathbf{V} = \mathbf{0}$ , then  $1 + \mathbf{U}^\top W^{-1}\mathbf{V} \neq 0$ . Therefore,  $\mathbf{V} \neq \mathbf{0}$ .

Since  $1 + \mathbf{U}^\top W^{-1}\mathbf{V} = 0$ ,  $\tilde{W}W^{-1}\mathbf{V} = \mathbf{V} + \mathbf{V}\mathbf{U}^\top W^{-1}\mathbf{V} = \mathbf{V} - \mathbf{V} = \mathbf{0}$ .

This contradiction completes the proof. 

Therefore, to calculate  $\tilde{W}^{-1}$  one needs to know  $W^{-1}$ . This approach may be beneficial in the applications, where  $W^{-1}$  is known or  $W$  is ‘easily invertible’.

In SAMRPR, the constraint matrix  $A$  is full rank, therefore  $AA^\top$  is non-singular.

## 4 Sherman–Morrison formula in IPM

### 4.1 IPM implementation using Sherman–Morrison formula

Recall that in IPM one has to solve the system

$$\begin{bmatrix} \nabla^2[\mathbf{t}\mathbf{f}(\mathbf{x}) + \varphi(\mathbf{x})] & A^\top \\ A & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \mathbf{v}^* \end{bmatrix} = \begin{bmatrix} -\nabla[\mathbf{t}\mathbf{f}(\mathbf{x}) + \varphi(\mathbf{x})] \\ \mathbf{0} \end{bmatrix}, \quad (14)$$

where  $\nabla^2[\mathbf{t}\mathbf{f}(\mathbf{x}) + \varphi(\mathbf{x})] = \mathbf{t}\mathbf{M} + \mathbf{I} + \mathbf{E}$  and  $\mathbf{E} = \text{diag}(1/x_i^2 - 1)$ .

In most cases, SAMRPR only requires non-negativity box-constraints. Then the system matrix is constructed as follows (for more general box-constraints the

procedure is the same, but more constraints have to be taken into account):

$$\begin{bmatrix} \nabla^2(\mathbf{t}\mathbf{f}(\mathbf{x}) + \varphi(\mathbf{x})) & \mathbf{A}^\top \\ \mathbf{A} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{A}^\top \\ \mathbf{A} & \mathbf{0} \end{bmatrix} + \sum_{i=1}^n \mathbf{V}_i \mathbf{V}_i^\top + \sum_{i=1}^n \mathbf{P}_i \mathbf{Q}_i^\top,$$

where

- $\mathbf{V}_i \in \mathbb{R}^{n+m}$ , all the elements of  $\mathbf{V}_i = (\mathbf{v}_1^i, \dots, \mathbf{v}_{m+n}^i)^\top$  are zeros except  $\mathbf{v}_i^i = \sqrt{\mathbf{t}\mathbf{m}_i}$ , where  $\mathbf{m}_i$  is the  $i$ th diagonal element of  $\mathbf{M}$ ;
- $\mathbf{P}_i \in \mathbb{R}^{n+m}$ , all the elements of  $\mathbf{P}_i = (\mathbf{p}_1^i, \dots, \mathbf{p}_{m+n}^i)^\top$  are zeros except  $\mathbf{p}_i^i = (1 - \mathbf{x}_i)/\mathbf{x}_i$ ;
- $\mathbf{Q}_i \in \mathbb{R}^{n+m}$ , all the elements of  $\mathbf{Q}_i = (\mathbf{q}_1^i, \dots, \mathbf{q}_{m+n}^i)^\top$  are zeros except  $\mathbf{q}_i^i = (1 + \mathbf{x}_i)/\mathbf{x}_i$ .

In order to solve (14) one needs to calculate

$$\mathbf{B} = \begin{bmatrix} \mathbf{I} & \mathbf{A}^\top \\ \mathbf{A} & \mathbf{0} \end{bmatrix}^{-1}, \quad (15)$$

and apply Sherman–Morrison formula  $n$  times (in pairs):

$$\tilde{\mathbf{W}} = \mathbf{B} + \sum_{j=1}^n (\mathbf{V}_j \mathbf{V}_j^\top + \mathbf{P}_j \mathbf{Q}_j^\top), \quad (16)$$

for each  $\mathbf{t}$ . At each step of this procedure the updated matrices are invertible, since all the diagonal elements of the top left block of matrix  $\mathbf{W}$  are positive. Therefore,  $1 + \mathbf{U}^\top \mathbf{W}^{-1} \mathbf{V} \neq 0$  at each step (break-free process). Each pair update  $\mathbf{V}_j \mathbf{V}_j^\top + \mathbf{P}_j \mathbf{Q}_j^\top$  affects only one element of  $\mathbf{B}$ , namely, the element  $(j, j)$ . Aggregation of these two updates in a single pair update allows one to reduce the computational time.

Now consider  $\mathbf{B}$  from (15). Suppose that  $\mathbf{B}$  has the following structure

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_1 & \mathbf{B}_2^\top \\ \mathbf{B}_2 & \mathbf{B}_3 \end{bmatrix}, \quad (17)$$

where  $\mathbf{B}_1 \in \mathbb{R}^{n \times n}$  has the same dimension as  $\mathbf{I}$  in (15),  $\mathbf{B}_2 \in \mathbb{R}^{m \times n}$  has the same dimension as  $\mathbf{A}$ , and  $\mathbf{B}_3 \in \mathbb{R}^{m \times m}$  has the same dimension as  $\mathbf{0}$  from (15). Then

$$\begin{cases} \mathbf{B}_1 + \mathbf{A}^\top \mathbf{B}_2 = \mathbf{I}; \\ \mathbf{A} \mathbf{B}_1 = \mathbf{0}; \\ \mathbf{B}_2^\top + \mathbf{A}^\top \mathbf{B}_3 = \mathbf{0}; \\ \mathbf{A} \mathbf{B}_2^\top = \mathbf{I}. \end{cases} \quad (18)$$

Finally,  $\mathbf{B}_1 = \mathbf{I} - \mathbf{A}^\top (\mathbf{A} \mathbf{A}^\top)^{-1} \mathbf{A}$ ,  $\mathbf{B}_2 = (\mathbf{A} \mathbf{A}^\top)^{-1} \mathbf{A}$  and  $\mathbf{B}_3 = -(\mathbf{A} \mathbf{A}^\top)^{-1}$ . If all the rows of  $\mathbf{A}$  are linearly independent, then

$$\text{rank}(\mathbf{A} \mathbf{A}^\top) = \text{rank}(\mathbf{A}^\top \mathbf{A}) = \text{rank}(\mathbf{A}) = \text{rank}(\mathbf{A}^\top) = m.$$

Thus  $\mathbf{A} \mathbf{A}^\top$  is a full rank square matrix and its inverse  $(\mathbf{A} \mathbf{A}^\top)^{-1} \in \mathbb{R}^{m \times m}$  exists. The main question now is how to find  $(\mathbf{A} \mathbf{A}^\top)^{-1}$ . Since  $m < n$  this problem is not as hard as the inverse in the original problem. The factorization  $(\mathbf{A} \mathbf{A}^\top)^{-1}$  can be found by applying Cholesky decompositions and triangular inverse (section 3). Dongarra et al. [4] reported that the corresponding LAPACK subroutines are able to invert large matrices ( $25000 \times 25000$ ). This is not enough for SAMRP, but it is reasonable for SAMRPR and other applications.

According to (16), we also need to apply  $n$  pairs of updates. However, these updates are not so computationally expensive.

## 4.2 Matrix update implementation using Sherman–Morrison formula

Our main computational difficulty (section 4.1) is to obtain  $(\mathbf{A} \mathbf{A}^\top)^{-1}$ . Suppose that we know  $(\mathbf{A}_i \mathbf{A}_i^\top)^{-1}$  for the  $i$ th time period. Now we need to calculate  $(\mathbf{A}_{i+1} \mathbf{A}_{i+1}^\top)^{-1}$  for the next time period. Suppose that only one element  $\mathbf{a}_{pq}^i$  of  $\mathbf{A}_i$  has to be updated, namely,  $\mathbf{a}_{pq}^{i+1} = \mathbf{a}_{pq}^i + \delta$ . Then  $\mathbf{A}_{i+1} = \mathbf{A}_i + \chi \mathbf{y}^\top$ , where  $\chi \in \mathbb{R}^m$  is an  $m$ -dimensional column vector with all the coordinates equal zero, except  $\chi_p = \delta$ , and  $\mathbf{y} \in \mathbb{R}^n$  is an  $n$ -dimensional

column vectors with all the coordinated equal zero, except  $\mathbf{y}_q = 1$ . Therefore,

$$\begin{aligned} \mathbf{A}_{i+1}\mathbf{A}_{i+1}^T &= (\mathbf{A}_i + \mathbf{x}\mathbf{y}^T)(\mathbf{A}_i + \mathbf{x}\mathbf{y}^T)^T = (\mathbf{A}_i + \mathbf{x}\mathbf{y}^T)(\mathbf{A}_i^T + \mathbf{y}\mathbf{x}^T) \\ &= \mathbf{A}_i\mathbf{A}_i^T + \mathbf{x}\mathbf{y}^T\mathbf{A}_i^T + \mathbf{A}_i\mathbf{y}\mathbf{x}^T + \mathbf{x}\mathbf{y}^T\mathbf{y}\mathbf{x}^T. \end{aligned}$$

Hence,  $(\mathbf{A}_{i+1}\mathbf{A}_{i+1}^T)^{-1}$  is calculated by applying the Sherman–Morrison formula three times:

1.  $\mathbf{V} = \mathbf{x}$ ,  $\mathbf{U}^T = \mathbf{y}^T\mathbf{A}_i$ ;
2.  $\mathbf{V} = \mathbf{A}_i\mathbf{y}$ ,  $\mathbf{U}^T = \mathbf{x}^T$ ;
3.  $\mathbf{V} = \sqrt{(\mathbf{y}^T\mathbf{y})}\mathbf{x}$ ,  $\mathbf{U}^T = \sqrt{(\mathbf{y}^T\mathbf{y})}\mathbf{x}^T$ .

If several elements are to be updated, then the above procedure has to be applied more than once. During the process ensure that the denominator in (13) is not zero.

### 4.3 Advantages of solving linear systems through matrix inverse in SAMRPR

In many cases solving linear systems through calculating the corresponding matrix inverse is not a very efficient approach. However, in this application (SAMRPR and SAMRP) the matrix inverse based approach together with the usage of Sherman–Morrison formula allows one to

- calculate an update for the matrix inverse appearing at different iterations of IPM (inside one time-period);
- update  $(\mathbf{A}\mathbf{A}^T)^{-1}$  when moving to the next time period.

Overall, the algorithm for solving SAMRPR over several periods of time, contains three steps.

1. Calculate the inverse of  $AA^T$ . This can be done in parallel (Cholesky decomposition and Triangular inverse). Use  $(AA^T)^{-1}$  and Sherman–Morrison formula to calculate the inverse of the matrix, appearing in IPM (first iteration).
2. Use Sherman–Morrison formula to calculate the inverses of the matrix updates, appearing in IPM at every iteration (same time-period).
3. Use Sherman–Morrison formula to update  $(AA^T)^{-1}$  for the next time-period. In this case, one needs to make sure that at every matrix update the final matrix is non-singular.

## 5 Numerical experiments

This section considers one time period for SAMRPR and applies the described matrix inverse approach to solve linear systems, appearing in IPM. We compare the results, obtained by calculating matrix inverses using the corresponding Matlab command and Sherman–Morrison formula. The main goal of this section is to test the computational time and robustness of the matrix inverse calculation procedure based on the Sherman–Morrison formula.

The results of numerical experiments are presented in Tables 1 and 2. These tables compare the results of numerical experiments with nearly identical core codes. The only difference between these algorithms is that in the case of “Default Inverse” the inverse matrix (15) was calculated by the corresponding Matlab command, while in the case of “Sherman–Morrison-based” this matrix was calculated using our Sherman–Morrison-based approach.

In Table 1, the column “Final obj. fun.” gives the final objective function value, and “Time ratio” lists the ratio of the corresponding computational time, namely, if this ratio is lower than one, then the Sherman–Morrison based approach is faster. In Table 2, the heading “Linear violation” corresponds to the linear constraint violation (Euclidian norm of the corresponding error



Table 1: Default inverse and Sherman–Morrison-based inverse: time and value.

Dimension	Defaults Inverse	Sherman–Morrison	Time ratio
	Final obj. fun.	Final obj. fun.	
$50 \times 100$	23.6	29.9	0.93
$50 \times 200$	44.9	46.1	0.97
$50 \times 500$	106.6	106.8	0.89
$50 \times 1000$	203.1	208.0	0.93
$100 \times 200$	44.9	47.8	0.95
$100 \times 500$	106.6	108.5	0.96
$100 \times 1000$	203.1	209.7	1.16

Table 2: Default inverse and Sherman–Morrison-based inverse: constraint violation.

Dimension	Defaults inverse violation		Sherman–Morrison violation	
	Linear	Box	Linear	Box
$50 \times 100$	3.36	−0.44	4E−16	feasible
$50 \times 200$	3.07	−0.46	3E−16	feasible
$50 \times 500$	3.10	−0.49	5E−16	feasible
$50 \times 1000$	3.34	−0.51	7E−15	feasible
$100 \times 200$	3.24	−0.46	5E−16	feasible
$100 \times 500$	3.27	−0.49	5E−16	feasible
$100 \times 1000$	3.57	−0.52	1E−14	feasible

vector  $\|Ax - \mathbf{b}\|$ ) and “Box violation” corresponds to the box-constraint violation: minimal value among the negative coordinates or “feasible” if all the coordinates are positive.

Observe from Table 1 that the computational time is better for the Sherman–Morrison-based approach (except the last one). The final objective function value is better for the Default Inverse approach. However, from Table 2 one can see that these better objective function values were reached on infeasible points. Moreover, most of the final results, obtained in the case of the Defaults Inverse approach are infeasible (both, linear and box-constraints violation). It is possible that a more precise choice of the internal parameters (for example, in the backtracking search subroutine) may allow one to obtain feasible results, but in this case the computational time will be increased significantly. Therefore, I conclude that the Sherman–Morrison-based approach is much more robust (slightly faster and much more stable) than the Default Inverse one, especially in the case of high dimensions.

## 6 Conclusions and further research directions

We show that our implementation of IPM is efficient for solving these problems. The main obstacle in our computation experiments is the repeated construction of large dimensional matrix inverses. We demonstrate that our Sherman–Morrison-based inverse calculation is slightly faster and more robust compared to the default inverse used in Matlab. In particular, our approach is much more stable than the default one. Also, it has been demonstrated that the proposed procedure is break-free, since  $1 + \mathbf{U}^T \mathbf{W}^{-1} \mathbf{V} \neq 0$ .

In our experiments, we only worked with matrices which are much smaller than the real matrices used in SAMRP or SAMRPR. The main reason is Matlab’s computational time. In the future we plan to develop a more elaborated procedure with parallelisation, which can be applied to a real SAMRPR or even SAMRP.

The Sherman–Morrison-based Inverse computation will be used in our future studies due to two main reasons. First, it is very efficient, fast and computationally stable. Second, it can be efficiently parallelised (calculation of  $AA^T$  through Cholesky decomposition and Triangular inverse), which is essential for this type of applications.

In many applications, the repeated solving of linear systems through the corresponding matrix inverse is not a very efficient approach. However, for SAMRPR and some other application this approach is very efficient, since it allows one to update the corresponding matrices inside IPM and also when moving to the next time-period. In this case, one needs to make sure that at every matrix update the final matrix is non-singular. This is a very important issue, which will be addressed in our future studies.

## References

- [1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, third edition, 1999. **E5**
- [2] M. S. Bartlett. An inverse matrix adjustment arising in discriminant analysis. *Ann. Math. Statist.*, 22(1):107–111, 03 1951. doi:[10.1214/aoms/1177729698](https://doi.org/10.1214/aoms/1177729698) **E11**
- [3] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2010. **E7**
- [4] Jack Dongarra, Mathieu Faverge, Hatem Ltaief, and Piotr Luszczek. High performance matrix inversion based on lu factorization for multicore architectures. In *Proceedings of the 2011 ACM international workshop on Many task computing on grids and supercomputers*, pages 33–42. ACM, 2011. **E5**, **E14**

- [5] G. E. Forsythe, M. A. Malcolm, and C. B. Moler. *Computer Methods for Mathematical Computations*. Prentice-Hall, Englewood Cliffs, N.J., 1977. Cited in Åke Björck's bibliography on least squares, <ftp://math.liu.se/pub/references> E5
- [6] Gene H. Golub and Robert J. Plemmons. Large-scale geodetic least-squares adjustment by dissection and orthogonal decomposition. *Linear Algebra and Its Applications*, 34:3–28, 1980. doi:[10.1016/0024-3795\(80\)90156-1](https://doi.org/10.1016/0024-3795(80)90156-1) E3
- [7] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 1996. E5, E10
- [8] Jacek Gondzio and Andreas Grothey. Exploiting structure in parallel implementation of interior point methods for optimization. Technical report, School of Mathematics, University of Edinburgh, Edinburgh, 2004. E4
- [9] Magnus R. Hestenes. Multiplier and gradient methods. *J. Optimization Theory Appl.*, 4:303–320, 1969. doi:[10.1007/BF00927673](https://doi.org/10.1007/BF00927673) E5
- [10] Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Standards*, 49:409–436, 1952. <http://www.sam.math.ethz.ch/~mhg/Latsis2002/jr49-6.pdf> E5
- [11] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984. doi:[10.1145/800057.808695](https://doi.org/10.1145/800057.808695) E5
- [12] Seung-Jean Kim, Kwangmoo Koh, Michael Lustig, Stephen Boyd, and Dimitry Gorinevsky. An interior-point method for large-scale L1-regularized least squares. *Selected Topics in Signal Processing, IEEE Journal of*, 1(4):606–617, 2007. doi:[10.1109/JSTSP.2007.910971](https://doi.org/10.1109/JSTSP.2007.910971) E3, E5

- [13] Yurii Nesterov and Arkadii Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*, volume 13 of *SIAM Studies in Applied mathematics*. 1993. E5
- [14] J. Nocedal and S. Wright. *Numerical Optimization*. Springer-Verlag, Inc., New York, 2006. E5
- [15] M. J. D. Powell. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, *Optimization*, pages 283–298. New York, NY, 1969.  
<http://www.ams.org/mathscinet-getitem?mr=42:7284> E5
- [16] Sherman Robinson, Andrea Cattaneo, and Moataz El-Said. Updating and estimating a social accounting matrix using cross entropy methods. TMD discussion papers 58, International Food Policy Research Institute, 2000. <http://ideas.repec.org/p/fpr/tmddps/58.html> E3
- [17] Jack Sherman and Winifred J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *Ann. Math. Statistics*, 21:124–127, 1950. doi:[10.1214/aoms/1177729893](https://doi.org/10.1214/aoms/1177729893) E5, E11

## Author address

1. **Nadezda Sukhorukova**, Faculty of Science, Engineering and Technology, Swinburne University of Technology, PO Box 218, Hawthorn, Vic 3122, Australia; and Faculty of Science and Technology, Federation University, PO Box 663, Ballarat, Vic 3353, Australia.  
<mailto:nsukhorukova@swin.edu.au>