# Computational Mathematics for Learning and Data Analysis

# Interior Points Method

Javad Khalili
Matricola : 546677

December 5, 2020

# 1 Introduction

Here we try to implement an algorithm for optimizing an objective function with the following structure:

$$(P) \quad min \quad f(x) = x^T Qx + q^T x, \tag{1a}$$

$$s.t. \sum_{i \in I^h} x_i = 1, \tag{1b}$$

$$x \geq 0 \tag{1c}$$

where,

- $Q \in R^{n \times n}$ is a symmetric positive semi-definite matrix
- $x \in R^n$ is the variable in which minimization is done
- $q \in R^n$ is a random $n \times 1$ vector
- The index sets $I^h, i = 1, ..., k$ form a partition of the set $1, ..., n$ for putting under consideration that vector $x$ is set of $k \in N$ disjoint simplices

However we will change the representation of (1b) to see how we have to consider this constraint to use while we try to solve the problem.
To solve this problem we use the primal-dual barrier approach[1] which compute solutions for a sequence of barrier problems for a decreasing sequence of barrier parameter $\mu$ converging to $zero$. Barrier function of problem (P) is strictly convex if (P) is convex and this implies that the barrier problem has at most one global minimum, and this global minimum, if it exists, is completely characterized by the KKT conditions(4).
Since in each iteration of this method we have to factorized the KKT matrix(Section 2.5.2) which is part of Newton's Method solution, computational complexity of $O(MaxIter \times \frac{(n+k)^3}{3})$ is expected.

## 2 Solution To The Problem

### 2.1 Notations

- $n \in N$ is the number of elements in vector $x$ which defines the dimension of matrix $Q$ and vector $z$ also.
- $k \in N$ is the number of disjoint simplices.
- $X \in R^{n*n}$ is diagonal matrix of vector $x.(diag(x))$
- $z \in R^{n*1}$ such that $z > 0$ is Lagrangian multiplier for bound constraint.
- $Z \in R^{n*n}$ is the diagonal matrix of vector $z.(diag(z))$
- $e \in R^{n*1}$ is a $n$-dimension vector of ones.

- $\mu$ is the barrier parameter
- $b \in R^{k*1}$ is a $k$-dimensions vector of ones.
- $\lambda \in R^{k*1}$ such that $\lambda > 0$ is the Lagrangian multiplier for equality constraint.
- $E \in R^{k*n}$ is full row rank matrix such that there is only one nonzero element (which is *one*) in each column, which will be chosen randomly in our implementation. this matrix help us to construct $k$ disjoint simplices that required to be consider in order to solve the problem.

  Example:
  We assume that $n = 5$ and $k = 2$, so we will have $x \in R^{5*1}$ and $E \in R^{k*n}$ such that there is one nonzero element(i.e. 1) in each column of $E$ matrix. suppose $E$ matrix is as follow:

  $$E = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \text{ and } x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

  since $k = 2$ there will be 2 dijoint simplices $I^{(1)}$ and $I^{(2)}$. Based on matrix $E$ the member of each simplex is as following:
  $I^{(1)} = \{x_1, x_2, x_3\}$ and $I^{(2)} = \{x_4, x_5\}$
  According to equality constrain(1b) the sum of items in each simplex must be equal to 1. so, $\sum_{i \in I^1} x_i = 1$ for $\{x_1, x_2, x_3\}$ and also $\sum_{i \in I^2} x_i = 1$ for $\{x_4, x_5\}$.
  As you can see there will a vector of ones with $k$-dimensions which call it $b$ that is result of multiplication of matrix $E$ and vector $x$.
  As mentioned before another representation of our equality constraint (1b) will be shown through out the solution which is going to be as following:

  $$Ex = b$$

## 2.2 Input Parameters

The input parameters will be provided in form of a structure to the main algorithm. All the elements that has been shown in previous subsection (Notations) will be considered as input parameters but $\mu$ that is going to be calculated inside each iteration in algorithm.

## 2.3 Primal-Dual Barrier Approach

The Lagrangian dual problem corresponding to our problem is another quadratic problem given by:

$$(D) \quad max \quad d(x, \lambda) = b^T \lambda - x^T Qx \quad , \tag{2a}$$

$$s.t. \quad E^T \lambda - Qx + z = q, \quad z \geq 0 \tag{2b}$$

If problem (P) has an optimal solution $x^*$ then there exist $\lambda^*$ and $z^*$ such that the point $(x^*, \lambda^*, z^*)$ is an optimal solution of problem (D). Conversely, if problem (D) has an optimal solution then problem (P) has an optimal solution. Moreover, the optimal solution values of both problems are identical. It is well-known that for all $x$ and $\lambda$ that are feasible for (P) and (D) we have

$$d(x, \lambda) \leq opt \leq f(x)$$

where *opt* denotes the optimal objective value for (P).
Optimality holds if and only if the complementary slackness relation

$$x^T z = 0$$

is satisfied.

We consider the logarithmic barrier function

$$min \quad f_\mu(x) = x^T Qx + q^T x - \mu \sum_{i=1}^{n} log(x_i), \tag{3a}$$

$$s.t. \quad Ex - b = 0 \tag{3b}$$

for decreasing sequence of positive Barrier parameter $\mu > 0$ converging to *zero.*

the first and second order derivation of $f_\mu$ are

$$\bigtriangledown f(x, \mu) = Qx + q - \mu X^{-1} e$$

$$\bigtriangledown^2 f(x, \mu) = Q + \mu X^{-2}$$

Consequently, $f$ is strictly convex on the relative interior of the feasible set. thus it achieves a minimum value at a unique point.and that unique point is characterized by the Karush-Kuhn-Tucker stationary condition.

## 2.4 KKT Conditions

Finding an optimal solution of Primal and Dual is equivalent to solving $KKT$ conditions which are as follows:

$$Qx + q + E^T\lambda - z = 0, \quad z \geq 0 \tag{4a}$$

$$Ex - b = 0, \quad x \geq 0 \tag{4b}$$

$$XZe = 0 \tag{4c}$$

where $Z$ is diagonal matrix having diag$(Z) = z$ and $e$ is $n$-dimensional ones vector.
Replacing the third equation , by the parameterized equation $XZe = \mu e$, the central path for QP is defined by the solution set $(x(\mu), \lambda(\mu), z(\mu)), \mu > 0$ of the following system.

$$Qx + q + E^T\lambda - z = 0 \tag{5a}$$

$$Ex - b = 0 \tag{5b}$$

$$XZe = \mu e \tag{5c}$$

## 2.5 Newton's Method

The present method computes an approximate solution to the barrier problem for a fix value of $\mu$ that decrease the barrier parameter and continue the solution of the next barrier problem from the approximate solution of the previous one. In order to solve the barrier problem for a given fixed value $\mu$ of the barrier parameter a Newton's method is applied to the primal-dual equations.
Here $j$ denotes the iteration counter for the inner loop.Given an iterate $(x_j, \lambda_j, z_j)$ with $x_j, z_j > 0$ ,search directions $(\Delta x_j\ ,\ \Delta\lambda_j\ ,\ \Delta z_j)$ obtained from linearization of KKT conditions at $(x_j, \lambda_j, z_j)$.
We obtain the linear system (5)

$$\begin{bmatrix} -Q & E^T & -I \\ E & 0 & 0 \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x_j \\ \Delta\lambda_j \\ \Delta z_j \end{bmatrix} = \begin{bmatrix} Qx + q - E^T\lambda - z \\ b - Ex \\ XZe - \mu e \end{bmatrix} \tag{6}$$

where $Q$ is the Hessian $\bigtriangledown^2_{xx} \quad L(x_j, \lambda_j, z_j)$ of the Lagrangian function for the original problem and $Z = \mu X^{-1}$, which can be written as $ZX = \mu e$. Instead of solving the non-symmetric linear system directly , the proposed method computes the solution equivalently by first the smaller, symmetric linear system

$$\begin{bmatrix} -(Q + X^{-1}Z) & E^T \\ E & 0 \end{bmatrix} \begin{bmatrix} \Delta x_j \\ \Delta\lambda_j \end{bmatrix} = \begin{bmatrix} Qx + q - \mu X^{-1}e - E^T\lambda \\ b - Ex \end{bmatrix} \tag{7}$$

and the vector $\Delta z_k$ is then obtained from

$$\Delta z_k = \mu_j X^{-1}e - z_k - X^{-1}Z\Delta x_k$$

We have to ensure that the matrix $Q+X^{-1}Z$ is positive definite. For this we can use Cholesky factorization to check whether $Q+X^{-1}Z$ is positive definite or not.

### 2.5.1 Solving System (7)

we consider linear system of (7) such that

$$\underbrace{\begin{bmatrix} -(Q + X^{-1}Z) & E^T \\ E & 0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \Delta x_j \\ \Delta\lambda_j \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} Qx + q - \mu X^{-1}e - E^T\lambda \\ b - Ex \end{bmatrix}}_{b} \tag{8}$$

since $Q$ is at least positive semi definite and we add positive diagonal matrix $X^{-1}Z$ to it, this matrix is non singular.Also since matrix $Q$ is symmetric, $A$ is also symmetric and indefinite(many zero eigenvalues). The most effective strategy to find solution in this case is to use a symmetric indefinite factorization [2], because of indefiniteness of matrix $A$ we cannot use the Cholesky factorization.

### 2.5.2 Symmetric Indefinite Factorization

Applying symmetric indefinite factorization on matrix $A$ of system (8) will have the following form:

$$PAP^T = LBL^T \tag{9}$$

Where $L$ is a lower triangular matrix , $B$ is a $1 \times 1$ or $2 \times 2$ block diagonal matrix and $P$ is a permutation matrix. To find the solution($x$) for system (8) by this approach we have to do series of computation which are going to be as following:
first we have to compute $P^T * b$ and result can be stored in some variable let's say $d$, then we have to compute $L^{-1}d$ to $e$ , and next $B^{-1}e$ to $f$ , next $L^{-T}f$ to $g$ and finally $x = P * g$. It can be shown as following too:

$$x = P\,\underbrace{L^{-T}\,\underbrace{B^{-1}\,\underbrace{L^{-1}\,\underbrace{P^T b}_{d}}_{e}}_{f}}_{g}$$

For implementing this in Matlab we can use $ldl()$ factorization Matlab function such that:

$$[L, B, P] = ldl(A);$$

and to solve $Ax = b$ system

$$x = P * (L' \backslash (B \backslash (L \backslash (P' * b))));$$

The time complexity of factorization (9) is $n^3/3$ flops and once it's computed we can solve the system(8)($Ax = b$) with $O(n^2)$.
To perform symmetric indefinite factorization, lets $A^{(t)}$ be the matrix that remains to be factorized in the $t$-th iteration. The algorithm starts with $A^{(t)} = A$. For each iteration first the sub-matrix $B^{(t)}$ will be identified from elements of $A^{(t)}$ that are suitable to be used as pivot block. There are many methods for selecting a suitable pivot $B^{(t)}$. For example Matlab uses MA57 routines to do the factorization and since MA57 uses Bunch-Parlett pivoting strategy which searches the submatrix at each stage for the largest magnitude diagonal $A^{(t)}_{qq}$ and the largest-magnitude off-diagonal $A^{(t)}_{rl}$. It identifies $A^{(t)}_{qq}$ as the $1 \times 1$ pivot block if the resulting growth rate is acceptable, otherwise it selects $\begin{bmatrix} A^{(t)}_{ll} & A^{(t)}_{lr} \\ A^{(t)}_{rl} & A^{(t)}_{rr} \end{bmatrix}$ as the $2 \times 2$ pivot block. Next $P^{(t)}$ should be found satisfying

$$(P^{(t)})^T A^{(t)} P^{(t)} = \begin{bmatrix} B^{(t)} & (C^{(t)})^T \\ C^{(t)} & Z^{(t)} \end{bmatrix}$$

The right hand side of above equation can be factorized as:

$$(P^{(t)})^T A^{(t)} P^{(t)} = \begin{bmatrix} I & 0 \\ C^{(t)}(B^{(t)})^{-1} & I \end{bmatrix} \cdot \begin{bmatrix} B^{(t)} & 0 \\ 0 & Z^{(t)} - C^t(B^{(t)})^{-1}(C^{(t)})^T \end{bmatrix} \cdot \begin{bmatrix} I & (B^{(t)})^{-1}(C^{(t)})^T \\ 0 & I \end{bmatrix}$$

let $L^{(t)} = C^{(t)}(B^{(t)})^{-1}$ and $A^{(t+1)} = Z^{(t)} - C^{(t)}(B^{(t)})^{-1}(C^{(t)})^T$ then the above equation can rewritten as:

$$(P^{(t)})^T A^{(t)} P^{(t)} = \begin{bmatrix} I & 0 \\ L^{(t)} & I \end{bmatrix} \cdot \begin{bmatrix} B^{(t)} & 0 \\ 0 & A^{(t+1)} \end{bmatrix} \cdot \begin{bmatrix} I & (L^{(t)})^T \\ 0 & I \end{bmatrix}$$

The same process can be repeated recursively on the matrix $A^{(t+1)}$. The dimension of $A^{(t+1)}$ is less than the dimension of $A^{(t)}$ by either one or two depending on the dimension of $B^{(t)}$ and $P$ is defined as a product of the permutation matrices from each iteration of the factorization.
It should be mentioned that since the matrix $E$ is a block diagonal, It could be exploited because of it's specific structure for sparser factorization by reordering the coefficient matrix in Newton's method linear system. In this implementation unfortunately I could not find a way to reorder the coefficient matrix in such a way that this property could be exploited.

#### 2.5.3 Alternative Approach

An alternative to the direct factorization techniques is to use an iterative method to solve the system (8). The conjugate gradient(CG) method is not recommended for solving the system(8), because it can be unstable on system that are not positive definite. Better options are Krylove methods (for example : GMRES) for general linear of symmetric indefinite systems.

## 2.6 Choosing the Barrier Parameter $\mu$

Given an iterate $(x, \lambda, z)$, consider an interior method that computes primal-dual search direction by(4). The most common approach for choosing the barrier parameter $\mu$ that appears on the right hand side of (4) is to make it proportional to the current complementary value, that is,

$$\mu_{j+1} = \sigma \frac{x_j^T z_j}{n}$$

where $\sigma > 0$ is a *centering parameter* and $n$ denotes the number of variables.

#### 2.6.1 Computing centering parameter $\sigma$

For computing $\sigma$ Mehrotra's predictor-corrector method[3] has been used that determines the value of $\sigma$ by a affine step computation.
First we calculate an affine step $(\Delta x_{affine}, \Delta\lambda_{affine}, \Delta z_{affine})$ by setting $\mu = 0$ in (6) that becomes:

$$\begin{bmatrix} -Q & E^T & -I \\ E & 0 & 0 \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x_{affine} \\ \Delta\lambda_{affine} \\ \Delta z_{affine} \end{bmatrix} = \begin{bmatrix} Qx + q - E^T\lambda \\ b - Ex \\ XZe \end{bmatrix} \tag{10}$$

And then compute $\alpha_{x_{affine}}$ and $\alpha_{z_{affine}}$ to be the longest steplength that can be taken along the direction before violating the non-negativity conditions for $x$ and $z$ ($(x, z) \geq 0$) with an upper bound of 1. The formula in section (2.7) can be used for affine step computation.
Note that since coefficient matrix in system (6) and (10) is the same matrix, we need only do the factorization only once and use it for solving affine step and and step direction systems.
Now, we define $\mu_{affine}$ to be the value of complementarity that would be obtained by a full step, that is:

$$\mu_{affine} = \frac{(x + \alpha_{x_{affine}} \Delta x_{affine})^T (z + \alpha_{z_{affine}} \Delta z_{affine})}{n}$$

and set the centering parameter to be

$$\sigma = \left( \frac{\mu_{affine}}{\left(\frac{x^T z}{n}\right)} \right)^3 .$$

## 2.7 Computing Steplength

After the step $\Delta = (\Delta x, \Delta\lambda, \Delta z)$ has been determined, we compute primal and dual steplength and define the new iterate $(x_{j+1}, \lambda_{j+1}, z_{j+1})$ as

$$x_{j+1} = x_j + \alpha \Delta x \quad , \quad \lambda_{j+1} = \lambda_j + \alpha \Delta \lambda \quad , \quad z_{j+1} = z_j + \alpha \Delta z$$

where:

$$\alpha_{primal} = \beta \Big( max(1, \frac{-\Delta x_j}{x_j}) \Big)^{-1} \tag{11a}$$

$$\alpha_{dual} = \beta \Big( max(1, \frac{-\Delta z_j}{z_j}) \Big)^{-1} \tag{11b}$$

$$\alpha = min(\alpha_{primal}, \alpha_{dual}) \tag{11c}$$

for given $0 < \beta < 1$ (In this implementation $\beta = 0.995$).

## 2.8 Starting Point

As starting point for solver I used selecting starting point of LP approach in [1], in order to prevent potential failure of convergence. For this we can start by computing $\bar{x}, \bar{z}$ and $\bar{\lambda}$ such that:

$$\bar{x} = E^T (EE^T)^{-1} b,$$

$$\bar{\lambda} = (EE^T)^{-1} E \nabla f(x),$$

$$\bar{z} = \nabla f(x) - E^T \bar{\lambda}$$

Where $\bar{x}$ is vector of minimum norm that satisfies primal constraint ($Ex = b$) and , $\bar{z}$ and $\bar{\lambda}$ are vectors that satisfy dual constraint ($E^T \lambda + z = \nabla f(x)$) such that $\bar{z}$ has minimum norm.
$\bar{x}$ and $\bar{z}$ are computed such way that they satisfy linear constraints of primal and dual but yet not the sign constraints ($x > 0, z > 0$) because in general there will be non-positive elements in both vector which violate the constraints, for this reason we define:

$$\delta_x = max(-(1.5) min \quad \bar{x}, 0)$$

and

$$\delta_z = max(-(1.5) min \quad \bar{z}, 0)$$

and modify the vectors $\bar{x}$ and $\bar{z}$,

$$x^* = \bar{x} + \delta_x e,$$

$$z^* = \bar{z} + \delta_z e$$

where $e \in R^{n\times 1}$ vector of ones.
and to make sure the elements of $x$ and $z$ are not too close to zero we the following:

$$\bar{\delta_x} = \frac{{x^*}^T z^*}{2e^T z^*},$$
$$\bar{\delta_z} = \frac{{x^*}^T z^*}{2e^T x^*}$$

where $\bar{\delta_z}$ is the average size of elements of $x^*$, weighted by corresponding elements of $z^*$, and vice versa for $\bar{\delta_z}$, then we set the starting point as following:

$$x^0 = x^* + \bar{\delta_x} e,$$
$$\lambda^0 = \bar{\lambda},$$
$$z^0 = z^* + \bar{\delta_z} e$$

## 2.9 Stopping Criteria

The algorithm stops when it reaches to optimal solution of the problem or it stops when the number of iterations exceeds the number assigned as $MaxIter$ by the user.
The optimal solution requires primal and dual infeasibility to be less than $1e-6$, and relative duality gap to become less than $1e-8$. These can be computed by following:

$$primal - infeasibility = \frac{\|Ex - b\|}{\|b\| + 1}, \tag{12a}$$

$$dual - infeasibility = \frac{\|Qx + q - E^T\lambda - z\|}{\|q\| + 1}, \tag{12b}$$

$$relative - duality - gap = \frac{x^T Qx + q^T x - b^T\lambda}{|x^T Qx + q^T x| + 1}. \tag{12c}$$

## 2.10 Algorithm

The implementation algorithm is as follow:

**Algorithm 1:** Primal-Dual Interior Point Methods

```
1  Function PDIPM( $Struct$ , $MaxIter$):
2      Initial values and starting point define is structure that explained in
         section (2.8) along with $\sigma = 0.5$ as initial value for first iteration
3      while $true$ do
4          Check for optimality conditions by computing (12)
5          if (12) satisfied then
6              break;
7          Check for stopping condition
8          if $i > MaxIter$ then
9              break;
10         Compute $\mu$ explained in section (2.6)
11         Sovle system (10) to compute affine step
12         Compute $\mu_{affine}$
13         Compute $\sigma$ for next iteration
14         Compute system(6) for step direction
15         Compute the step length using (11)
16         Update the value of $x, z$ and $\lambda$ for next iteration
17         $x_{i+1} = x_i + \alpha \Delta x_i$
18         $\lambda_{i+1} = \lambda_i + \alpha \Delta \lambda_i$
19         $z_{i+1} = z_i + \alpha \Delta z_i$
20     return [$primal$ , $x$ , $status$]
```

## 2.11 Numerical Experiment

### 2.11.1 Generating Instances

For generating instances I used a Matlab structure that take two inputs from user ($n$ and $k$) to create corresponding size instances. For creating a semi definite matrix ($Q$) for objective function we can take a rectangular $m \times n$ matrix and multiply it by it's transpose. I considered $m = n - k$ so that the rectangular matrix let's say $G$ be a random $(n-k) \times n$ matrix, and $Q = G^T G$.
For the sparse instances I used Matlab *sprand*() function with density 0.01 non zero elements.
The following algorithm is for creating $E \in R^{k \times n}$:

**Algorithm 2:** Creating $E$ matrix

```
Result: $E$ matrix
$E$ = zeros($k$,$n$);
$counter = 1$;
$row = 1$;
while $row \leq k$ do
    $col = \lceil \frac{n}{k} \rceil$;
    $c = 1$;
    while $c \leq col$ do
        $E(row, counter) = 1$;
        $counter++$;
    end
    $n = n - col$;
    $k--$;
end
```

### 2.11.2 Tests

In the following two tables the results of executing the algorithm for different instances with different sizes and structures are shown such that each instance has been created 4 times and average result has been recorded in Table 1 for dense instances and Table 2 for sparse instances.
I also used a standard IP solver (*quadprog*) provided by Matlab to run the same instances on to make sure the solution is correct and see performance difference between Algorithm 1 and the solver.

| Instance | | *quadprog* | | *PDIPM* | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | nnz(Q) | iter | time(s) | iter | time(s) | Fact. time(s) | Predictor | Corrector | Rel.dual | Compl | norm |
| Prob_1_1 | 514 | 8.8 | 0.10 | 10.4 | 0.09 | 0.004 | 0.015 | 0.025 | 1.75e-9 | 1.75e-9 | 6.72e-9 |
| Prob_1_2 | 511 | 9.2 | 0.13 | 9.8 | 0.13 | 0.005 | 0.023 | 0.044 | 1.03e-10 | 1.34e-9 | 2.80e-7 |
| Prob_1_3 | 507 | 9 | 0.22 | 9,2 | 0.10 | 0.009 | 0.017 | 0.032 | 5.15e-10 | 1.98e-9 | 2.47e-7 |
| Prob_1_4 | 487 | 8.4 | 0.10 | 9 | 0.11 | 0.01 | 0.017 | 0.036 | 1.15e-10 | 1.98e-9 | 2.47e-8 |
| Prob_2_1 | 2560 | 9.8 | 0.19 | 12.2 | 0.51 | 0.115 | 0.127 | 0.279 | 2.04e-11 | 1.01e-9 | 2.62e-8 |
| Prob_2_2 | 2008 | 10 | 0.20 | 10.2 | 0.59 | 0.111 | 0.124 | 0.260 | 1.12e-11 | 1.03e-9 | 2.73e-8 |
| Prob_2_3 | 1840 | 10.2 | 0.24 | 9.8 | 0.78 | 0.164 | 0.165 | 0.337 | 1.12e-10 | 3.98e-9 | 1.14e-6 |
| Prob_2_4 | 1614 | 9.8 | 0.26 | 9.8 | 0.91 | 0.194 | 0.199 | 0.140 | 1.21e-11 | 1.04e-9 | 2.11e-7 |
| Prob_3_1 | 5305 | 10 | 0.48 | 14 | 3.05 | 0.568 | 0.675 | 1.473 | 2.94e-11 | 2.95e-9 | 4.33e-7 |
| Prob_3_2 | 5222 | 10.4 | 0.51 | 10.6 | 2.75 | 0.550 | 0.602 | 1.315 | 1.44e-11 | 1.56e-9 | 4.74e-8 |
| Prob_3_3 | 5302 | 10.6 | 0.72 | 10.8 | 3.98 | 0.876 | 0.875 | 1.908 | 2.02e-11 | 6.83e-10 | 6.20e-7 |
| Prob_3_4 | 5278 | 10.6 | 0.82 | 10.2 | 4.39 | 1.013 | 0.982 | 2.053 | 9.76e-11 | 2.03e-9 | 1.65e-6 |
| Prob_4_1 | 10605 | 10.8 | 1.73 | 14.2 | 14.83 | 3.35 | 3.204 | 6.902 | 3.80e-12 | 7.75e-10 | 5.94e-7 |
| Prob_4_2 | 10568 | 11 | 2.21 | 11.2 | 14.29 | 3.339 | 3.131 | 6.637 | 1.47e-11 | 3.15e-9 | 7.26e-7 |
| Prob_4_3 | 10574 | 10.8 | 3.01 | 10.4 | 18.88 | 4.642 | 4.221 | 8.714 | 1.04e-10 | 3.72e-9 | 3.45e-7 |
| Prob_4_4 | 10441 | 10.4 | 3.54 | 10.2 | 22.17 | 5.645 | 5.097 | 10.049 | 2.27e-12 | 2.16e-9 | 1.25e-7 |
| Prob_5_1 | 21141 | 11.2 | 10.34 | 14.2 | 9357 | 22.131 | 20.795 | 43.939 | 1.60e-11 | 4.86e-9 | 4.23e-7 |
| Prob_5_2 | 21170 | 10.6 | 12.47 | 11.4 | 92.18 | 22.31 | 20.735 | 43.438 | 1.78e-9 | 1.79e-9 | 3.20e-7 |
| Prob_5_3 | 21046 | 11 | 16.31 | 11 | 108.23 | 26.98 | 24.568 | 50.694 | 2.35e-11 | 6.07e-9 | 1.36e-7 |
| Prob_5_4 | 21192 | 10.8 | 23.72 | 10.8 | 151.62 | 38.791 | 35.589 | 70.108 | 4.92e-12 | 2.68e-9 | 3.14e-7 |
| Prob_1_1 | | | | | | | | | | | |
| Prob_1_2 | | | | | | | | | | | |
| Prob_1_3 | | | | | | | | | | | |
| Prob_1_4 | | | | | | | | | | | |
| Prob_2_1 | | | | | | | | | | | |
| Prob_2_2 | | | | | | | | | | | |
| Prob_2_3 | | | | | | | | | | | |
| Prob_2_4 | | | | | | | | | | | |
| Prob_3_1 | | | | | | | | | | | |
| Prob_3_2 | | | | | | | | | | | |
| Prob_3_3 | | | | | | | | | | | |
| Prob_3_4 | | | | | | | | | | | |
| Prob_4_1 | | | | | | | | | | | |
| Prob_4_2 | | | | | | | | | | | |
| Prob_4_3 | | | | | | | | | | | |
| Prob_4_4 | | | | | | | | | | | |
| Prob_5_1 | | | | | | | | | | | |
| Prob_5_2 | | | | | | | | | | | |
| Prob_5_3 | | | | | | | | | | | |
| Prob_5_4 | | | | | | | | | | | |

Table 1: Exprimetns on Sparse Matrices

| Instance | | | 1-corr | 2-corr | 3-corr | 4-corr | 5-corr | 6-corr |
|---|---|---|---|---|---|---|---|---|
| No. | Instance | nnz(Q) | iter | iter | iter | iter | iter | iter |
| 1 | Prob_1_1 | dense | | | | | | |
| 2 | Prob_1_4 | dense | | | | | | |
| 3 | Prob_2_1 | dense | | | | | | |
| 4 | Prob_2_4 | dense | | | | | | |
| 5 | Prob_3_1 | dense | | | | | | |
| 6 | Prob_3_4 | dense | | | | | | |
| 7 | Prob_4_1 | dense | | | | | | |
| 8 | Prob_4_4 | dense | | | | | | |
| 9 | Prob_1_1 | dense | | | | | | |
| 10 | Prob_1_4 | dense | | | | | | |
| 11 | Prob_2_1 | 496 | | | | | | |
| 12 | Prob_2_4 | 524 | | | | | | |
| 13 | Prob_3_1 | 2621 | | | | | | |
| 14 | Prob_3_4 | 2583 | | | | | | |
| 15 | Prob_1_1 | 5306 | | | | | | |
| 16 | Prob_1_1 | 5333 | | | | | | |
| 17 | Prob_1_1 | 10492 | | | | | | |
| 18 | Prob_1_1 | 10504 | | | | | | |
| 19 | Prob_1_1 | 21163 | | | | | | |
| 10 | Prob_1_1 | 20975 | | | | | | |

Table 2: Experiments for determining k correction

## References

[1] Jorge Nocedal, Stephen J. Wright "Numerical Optimization," Second Edition

[2] Glub G. H. and Loan C. F. V. " Matrix Computations," The Johns Hopkins University Press, Baltimore,Maryland, United States,2012.

[3] S. Mehrotra. "On the implementation of a primal-dual interior point method." SIAM Journal on Optimization,1992.