On the LVI-Based Numerical Method (E47 Algorithm) for Solving Quadratic Programming Problems

Yunong Zhang, Senbo Fu, Zhijun Zhang, and Lin Xiao
School of Information Science and Technology
Sun Yat-sen University
Guangzhou, 510006, China
ynzhang@ieee.org, fussenbo900@163.com

Xuezhong Li
School of Software
Sun Yat-sen University
Guangzhou, 510006, China
lixzh@student.sysu.edu.cn

Abstract—In this paper, a numerical method (termed, E47 algorithm) based on linear variational inequalities (LVI) is presented and investigated to solve quadratic programming (QP) problems which are simultaneously subject to linear equality, inequality and bound constraints. Note that such constrained QP problems can be equivalent to linear variational inequalities and then to piecewise-linear projection equations (PLPE). The E47 algorithm is then adapted to solving the resultant PLPE, and thus the optimal numerical solutions to the QP problems are obtained. In addition, the global linear convergence of such an E47 algorithm is proved. The numerical comparison results between such an E47 algorithm and the active set algorithm are further provided. The efficacy and superiority of the presented E47 algorithm for QP solving are substantiated.

Index Terms—Numerical algorithm, quadratic programming (QP), linear variational inequalities (LVI), global convergence.

I. INTRODUCTION

In numerous areas of application such as engineering and scientific research, manufacturing, and economic statistics, quadratic programming (QP) problems are widely encountered and investigated by researchers; e.g., [1]–[5]. Especially in recent years, different kinds of online solution methods for QP problems have been proposed and developed, e.g., recurrent neural networks (RNN) and numerical algorithms (NA) [2], [6]–[9]. In [6], two new classes of high-performance networks with economical analog multipliers are proposed for solving the QP problems. In [7], a simplified LVI-based primal-dual neural network (LVI-PDNN) is presented as a powerful tool for solving online the QP problems. However, in literature (e.g., [9], [10] and most textbooks), people often discuss the QP problems subject to one or two kinds of typical constraints (e.g., only the equality constraint and/or the inequality constraint). In actual manufacturing and economic areas, the QP problems are often simultaneously subject to all (or to say, three) kinds of constraints [11], i.e., the equality constraint, the inequality constraint and the bound constraint. It would observably increase the computational complexity if we simply extend and apply the existing methods (which are designed for solving the QP problems subject to one or two kinds of constraints) to solving the general QP problems with three kinds of constraints [7]. Thus, the existing methods may not be efficient enough on solving the general QP problems subject to three kinds of constraints and may lose the practical value. Motivated by engineering applications of the QP, e.g., in robotics [2], [12], we prefer the following general problem formulation:

\[-\frac{x^T W x}{2} + q^T x,\]

subject to \[J x = d,\] \[A x \leq b,\] \[\xi^- \leq x \leq \xi^+,\]

where \(x \in \mathbb{R}^n\) is the decision vector to be obtained; superscript \(\top\) denotes the transpose of a vector or matrix; and \(W \in \mathbb{R}^{n \times n}\) is assumed to be a positive-definite symmetric matrix. The coefficients are defined respectively as \(q \in \mathbb{R}^n, J \in \mathbb{R}^{m \times n}, d \in \mathbb{R}^m, A \in \mathbb{R}^{\dim(b) \times n}\) and \(b \in \mathbb{R}^{\dim(b)}\), with \(\dim(b)\) denoting the dimension of vector \(b\). Besides, \(\xi^-\) and \(\xi^+\) denote the lower and upper bounds of \(x\), respectively.

Before proceeding, let us review the existing QP solvers in relatively more detail, especially in the authors’ background of RNN. In existing literature, most QP solvers can be classified to be either parallel-processing methods such as neural networks or serial-processing numerical algorithms [7], [10], [11]. Some of existing neural networks contain penalty parameters, of which the stable equilibrium points (corresponding to the solutions of optimization problems) can be achieved only when the penalty parameters are infinitely large. This is almost impossible when we solve the problem numerically [6]. Ref. [10] proposes a low-order discrete-time recurrent neural network for solving high-dimension QP problems, especially for the case that the number of decision variables is close to the number of constraints. However, the QP problem depicted in [10] only involves the equality constraint. Some numerical algorithms are also proposed to solve the QP problem, e.g., the active set algorithm [13], the quasi-Newton method [14], and the Lagrange algorithm [15]. Based on Karush-Kuhn-Tucker optimality condition and the projection operator [7], researchers have improved the traditional dual neural network (DNN) [11], [12]. However,
the improved DNN still has the disadvantage; i.e., in the process of solving the QP problems, the inversion of coefficient matrix $W$ is required, and thus it can only handle strictly convex QP problems with fixed and/or easy-to-invert coefficient-matrix $W$ [7], [11], [12].

In order to solve general QP problems (subject to equality, inequality and bound constraints, simultaneously) and obtain optimal solutions for robot control, a numerical method (termed, E47 algorithm) based on linear variational inequalities (LVI) is developed, presented and investigated in this paper. Note that, for simplicity, we call it E47 algorithm which is named after equations (4)-(7) of [16] [which was designed originally to solve linear projection equations]. To do so, the general QP problems are firstly converted via the important “bridge” theorem into linear variational inequalities (LVI) [7], [17]. The E47 algorithm is then adapted to solving the resultant piecewise-linear projection equations, the LVI as well as the QP problems. Furthermore, the numerical comparison results between such an E47 algorithm and the conventional active-set algorithm are provided. The conducted numerical experiments demonstrate well the efficacy and superiority of such an E47 algorithm for QP solving. The remainder of this paper is organized into four sections. The important “bridge” theorem and the E47 algorithm are presented in Section II. Theoretical results and detailed proof of the E47 algorithm are presented in Section III. The global convergence behaviors are further investigated via numerical experiments in Section IV. Section V concludes this paper with final remarks. Before ending this section, we list the main novelties and contributions of the work as follows.

- This paper considers the QP problems subject to all kinds of linear constraints (i.e., equality, inequality and bound constraints) simultaneously. This is quite different from the conventional situation of handling one or two kinds of constraints in most literature and textbooks.

- Based on the authors’ research background of both RNN and NA, an LVI-based numerical method with a very simple and implementable structure is proposed to solve accurately the QP problems. This numerical algorithm can be viewed as an efficient discrete-time model of LVI-PDNN, which it takes researchers (including the authors) decades to finally find. Note that this interdisciplinary LVI-based numerical method is very novel and different from the results of pure RNN or NA research community.

- Before adapting and applying the E47 algorithm to the QP solving, an important “bridge” theorem has to be established, which is generalized from the authors’ previous work and connects QP, LVI and PLPE. The “bridge” theorem is unique and elegant, as it equivalently converts the general QP (1)-(4) to an LVI problem, without using dual vectors for bound constraint (4).

- The core equations (4)-(7) of [16] were originally designed for solving linear projection equations, but now they are adapted to solving QP problems. That is, the authors of this paper have discovered the new area and utility of such core equations.

- The global linear convergence of the E47 algorithm is presented for QP solving, with proof detailed according to the authors’ engineering-type understanding.

- Numerical experiments substantiate well the efficacy and superiority of the E47 algorithm for QP solving (it is worth mentioning that, on average, the E47 algorithm implemented in C program is 168 times faster than MATLAB built-in function “Quadprog”, and that the former is thus now applied in real-time robot control).

II. “BRIDGE” THEOREM AND E47 ALGORITHM

With the aid of dual decision variables, for the primal QP problem (1)-(4), its dual QP problem can be derived via duality theory [2], [7], [17]. For each constraint, such as (2), (3) and (4), the dual decision variables can be defined as the Lagrangian multipliers. In addition, for bound constraint (4), we use an elegant treatment to cancel the dual decision variables to reduce the QP-solver complexity. Thus we have the following “bridge” theorem [i.e., the conversion from QP (1)-(4) to linear variational inequality (LVI) (5)].

**Theorem 1.** Assume the existence of optimal solution $x^*$ to QP problem (1)-(4). QP (1)-(4) is then equivalent to the LVI problem which is to find a vector $y^* \in \Omega := \{y|\varsigma^\ominus \leq y \leq \varsigma^+\} \subset \mathbb{R}^{n+m+\dim(b)}$ such that
\[
(y - y^*)^T(Hy^* + p) \geq 0, \ \forall y \in \Omega,
\]
where the primal-dual decision vector $y$ and its lower and upper bounds are defined respectively as
\[
y = \begin{bmatrix} x \\ u \\ v \end{bmatrix}, \ \varsigma^- = \begin{bmatrix} \xi^- \\ -\varsigma v u \\ 0 \end{bmatrix}, \ \varsigma^+ = \begin{bmatrix} \xi^+ \\ +\varsigma v u \\ +\varsigma v v \end{bmatrix}.
\]

In addition, $1_v$ and $1_u$ denote appropriately-dimensioned vectors composed of ones, and $\varsigma$ is defined sufficiently large to replace $+\infty$ numerically. Vectors $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^{\dim(b)}$ are the dual decision vectors defined corresponding to equality constraint (2) and inequality constraint (3). The augmented coefficients in LVI (5) are defined as
\[
H = \begin{bmatrix} W & -J^T & A^T \\ J & 0 & 0 \\ -A & 0 & 0 \end{bmatrix}, \ p = \begin{bmatrix} q \\ -d \\ b \end{bmatrix}.
\]

**Proof.** Can be generalized from [2], [7], [17].

Secondly, it is well-known [7] that LVI (5) is equivalent to the following piecewise-linear projection equation (PLPE):
\[
P_\Omega(y - (Hy + p)) - y = 0,
\]
where piecewise-linear projection operator $P_\Omega(\cdot) : \mathbb{R}^{n+m+\dim(b)} \rightarrow \Omega = \{y|\varsigma^- \leq y \leq \varsigma^+\}$ is extensively
Thus be efficient for very large sparse problems. The main advantage of the E47 algorithm is that it possesses optimal exploitation of the sparsity of matrix $H$ of the E47 algorithm consists essentially of only two procedures: the first, $H^T y$ or $H y$, and the second procedure $d(y^k)$ with $d(y^k)$ chosen as $H^T e(y^k)$ or $H y$.

Theorem 2. For any $y^*$ belonging to $\Omega^*$, the sequence $\{y^k\}$ (with iteration index $k = 0, 1, 2, 3, \ldots$) generated by the E47 algorithm satisfies

$$||y^{k+1} - y^*||_2^2 \leq ||y^k - y^*||_2^2 - \rho(y^k) ||e(y^k)||_2^2.$$  

Proof. See [18] and can be generalized from it.

Theorem 3. The sequence $\{y^k\}$ generated by the E47 algorithm (7)-(9) converges to an optimal solution $y^*$.

III. Theoretical Results and Proof

In this section, by following [16], [18], the global linear convergence of E47 algorithm (7)-(9) is proved as below.

$$||y^{k+1} - y^*||_2^2 \leq ||y^k - y^*||_2^2 - \rho(y^k) ||e(y^k)||_2^2.$$  

Proof. For any $\Omega^*$, the sequence $\{y^k\}$ (with iteration index $k = 0, 1, 2, 3, \ldots$) generated by the E47 algorithm satisfies

$\forall i \in \{P^j(y)\}$. For graphical interpretation of the $i$th processing element $[P^j(y)]_i$, see Fig. 1.

Thirdly, solving LVI (5) is equivalent to finding a zero of the following error function which is actually the negation of the left-hand side of PLPE (6): $e(y) := y - P^j(y - (H y + p))$. The following numerical algorithm (i.e., E47 algorithm) is then employed to solve PLPE (6), LVI (5) and QP (1)-(4) as follows. Let the optimal solution set $\Omega^* = \{y^*\}$ or $\{y^*\}$ be a solution of (6). Given $y^b \in R^{n+m+dim(b)}$, for iteration index $k = 0, 1, 2, 3, \ldots$ (used as the superscript, e.g., superscript $^0$ in $y^0$), $\forall y^k \in \{y^*\}$, then we have the following E47 iteration formula:

$$y^{k+1} = P^j(y^k - \rho(y^k) d(y^k)),$$  

with $d(y^k) := H^T e(y^k) + H y^k + q,$

$$\rho(y^k) := \frac{||e(y^k)||_2^2}{||(H^T + I)e(y^k)||_2^2}.$$  

The main advantage of the E47 algorithm is that it possesses a very simple and implementable structure, in addition to the ability to handle the piecewise-linear projection equation (6) and QP (1)-(4) (which is subject to all kinds of linear constraints); whereas some other algorithms (e.g., [6], [8]) may only solve special cases of (6). Evidently, each iteration of the E47 algorithm consists essentially of only two projections to set $\Omega$ and only two matrix-vector products (i.e., $H^T e(y)$ and $H y$), whereas the E47 algorithm, generally of $O((n + m + dim(b))^2)$ operations, could also allow the optimal exploitation of the sparsity of matrix $H$ and may thus be efficient for very large sparse problems.

IV. Numerical-Experiment Results

To demonstrate the efficacy of E47 algorithm (7)-(9), we implement this algorithm via both MATLAB and C programming languages to solve general QP problems. The numerical experiments are carried out in MATLAB R2008a environment performed on a personal digital computer equipped with an INTEL Core(TM) Duo E4500 2.20GHz CPU, 2GB DDR3 memory, and a Windows XP Professional operating system. The final output errors and the computing time of algorithms are shown in this section.

A. QP problem with specified coefficients

In this subsection, without loss of generality, the following QP problem is considered with corresponding coefficient parameters specified.

$$\text{minimize } x^T W x/2 + q^T x,$$  

subject to $J x = d,$

$A x \leq b,$

$\xi^- \leq x \leq \xi^+,$
Throughout this paper, the prescribed error criterion $\|e(y_k)\|_2 \leq 10^{-3}$ is used, which is accurate for most actual applications (for example, in robotics, $10^{-3}$ in meters means that the high precision of less than 1 millimeter is achieved). To show the intermediate results and the convergence performance of E47 algorithm (7)-(9), the value of $x^k$ (which is a column vector made of the first $n$ elements of $y^k$) is tracked and depicted in Fig. 2. Specifically, two numerical experiments with different initial states are conducted to demonstrate the efficacy of this E47 algorithm, i.e., shown in Fig. 2 (a) and (b). As seen from the figure, starting from two different initial states, the E47 algorithm both converges to the same unique optimal solution. In addition, we see that it needs about 500 iterations to achieve the optimal solution of the QP problem (12)-(15) with high accuracy. Note that the optimal solution of QP problem (12)-(15) is $x^* = [2, 1, -6]^T$. Fig. 3 further illustrates the corresponding computational error, i.e., $\|x^k - x^*\|_2$ over $k$. From the figure, we see that $\|x^k - x^*\|_2$ converges to 0 as iteration index $k$ increases. Thus we can say that the E47 iteration process is convergent, and the E47 algorithm is successful in solving QP problem (12)-(15).

B. QP problems with randomly-generated coefficients

Following the above numerical experiments with coefficients specified, we further investigate the E47 algorithm for solving the QP problems with coefficients randomly generated. That is to say, the values of coefficients $W$, $q$, $J$, $d$, $A$, $b$, $\xi^-$, and $\xi^+$ are all randomly generated by using MATLAB function “rand()”. The prescribed error criterion $\|e(y_k)\|_2 \leq 10^{-3}$ is used again, being the same as before. For comparison and for illustration, we not only develop the MATLAB program for the E47 algorithm, but also implement the algorithm in C program. For these randomly-generated solvable quadratic-programming problems, the presented E47 algorithm in this paper shows great advantages. That is, we conduct numerous numerical experiments of E47 algorithm.
(7)-(9) using both MATLAB and C programs, and obtain desired results also in this random situation. The results about the average computing time and the final output errors of 20 experiments are shown in Table I. As seen from the table, the computing time of the E47 algorithm is relatively very small (i.e., the average one of the C program being 0.0044 s, in other words, 4.4 milliseconds). In addition, the final output errors are tiny (less than $10^{-3}$). Another observation from the table of these random numerical experiments is that the E47 algorithm implemented in the C program is usually much (around 8 times) faster than that implemented in the MATLAB program (i.e., via the M-file). In summary, the fast solution speed and the tiny computational errors substantiate the efficacy of the E47 algorithm presented in this paper.

C. Comparison between E47 and active-set algorithms

To better illustrate the advantages of the presented E47 algorithm, a series of general QP problems are solved and presented in this subsection via both E47 algorithm (7)-(9) and the active-set algorithm. The computing time and the solutions difference for solving the randomly-generated QP problems are recorded for both E47 and active-set algorithms. In addition, the comparison on the average computing time and the average solution-difference between the E47 algorithm and the active set algorithm is presented in Table II. Note that the active set algorithm adopted and tested here is actually a built-in function of MATLAB under the name of “Quadprog”. The 10 groups of comparison tests in the table (i.e., Table II) actually correspond to Tests 11 through 20 in Table I, where, without loss of generality, $n = 3$, $m = 1$, and dim(b) = 2.

As seen from Table II, the average computing time for solving a QP problem by E47 algorithm (MATLAB code) is about 0.04 s while the average computing time of the active set algorithm is about 0.84 s. These results demonstrate that the computing time of the E47 algorithm (MATLAB code) is about 21 times faster than that of the active set algorithm. So, we can estimate that the E47 algorithm implemented in the C program is $21 \times 8 = 168$ times faster than the MATLAB built-in function “Quadprog”. In summary, the numerical-comparison results substantiate well the efficacy and superiority of the presented E47 algorithm for QP solving. Compared to the active set algorithm, the E47 algorithm can achieve almost the same accuracy (i.e., solution effectiveness) in solving the randomly-generated solvable QP problems, which is shown in the fourth column of Table II.

<table>
<thead>
<tr>
<th>Test #</th>
<th>TimeMATLAB (s)</th>
<th>TimeC (s)</th>
<th>ErrorMATLAB</th>
<th>ErrorC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.02181373102</td>
<td>0.003994436004</td>
<td>8.90078534207947 $\times 10^{-4}$</td>
<td>8.90078534207018 $\times 10^{-4}$</td>
</tr>
<tr>
<td>2</td>
<td>0.01721218900</td>
<td>0.00348268049</td>
<td>7.33399349422728 $\times 10^{-4}$</td>
<td>7.33399349422764 $\times 10^{-4}$</td>
</tr>
<tr>
<td>3</td>
<td>0.02658747700</td>
<td>0.00338566885</td>
<td>8.75932512358256 $\times 10^{-4}$</td>
<td>8.75932512358256 $\times 10^{-4}$</td>
</tr>
<tr>
<td>4</td>
<td>0.02908979002</td>
<td>0.00399562775</td>
<td>7.40338441679319 $\times 10^{-4}$</td>
<td>9.63476282589444 $\times 10^{-4}$</td>
</tr>
<tr>
<td>5</td>
<td>0.0250408858</td>
<td>0.00392329257</td>
<td>9.10313617957184 $\times 10^{-4}$</td>
<td>9.10313617957253 $\times 10^{-4}$</td>
</tr>
<tr>
<td>6</td>
<td>0.07280702676</td>
<td>0.00652331398</td>
<td>9.74053797216939 $\times 10^{-4}$</td>
<td>9.7921354792625 $\times 10^{-4}$</td>
</tr>
<tr>
<td>7</td>
<td>0.01841975193</td>
<td>0.00359698746</td>
<td>6.57917212034314 $\times 10^{-4}$</td>
<td>7.57917215380154 $\times 10^{-4}$</td>
</tr>
<tr>
<td>8</td>
<td>0.05837775869</td>
<td>0.00513098081</td>
<td>9.65815016827337 $\times 10^{-4}$</td>
<td>9.53240506576691 $\times 10^{-4}$</td>
</tr>
<tr>
<td>9</td>
<td>0.02287355595</td>
<td>0.00379865711</td>
<td>6.8624374747358 $\times 10^{-4}$</td>
<td>8.91371467172357 $\times 10^{-4}$</td>
</tr>
<tr>
<td>10</td>
<td>0.18806828372</td>
<td>0.0035762189</td>
<td>9.83050001295021 $\times 10^{-4}$</td>
<td>9.83050001295021 $\times 10^{-4}$</td>
</tr>
<tr>
<td>11</td>
<td>0.02871047151</td>
<td>0.00474342544</td>
<td>9.70253520703793 $\times 10^{-4}$</td>
<td>9.70253520704082 $\times 10^{-4}$</td>
</tr>
<tr>
<td>12</td>
<td>0.01795118477</td>
<td>0.00387432559</td>
<td>9.47721317780562 $\times 10^{-4}$</td>
<td>9.47721317814435 $\times 10^{-4}$</td>
</tr>
<tr>
<td>13</td>
<td>0.03199583371</td>
<td>0.00422276974</td>
<td>9.6680602758962 $\times 10^{-4}$</td>
<td>9.38488935465161 $\times 10^{-4}$</td>
</tr>
<tr>
<td>14</td>
<td>0.04852938228</td>
<td>0.00517810318</td>
<td>9.25414059358433 $\times 10^{-4}$</td>
<td>9.58079766524444 $\times 10^{-4}$</td>
</tr>
<tr>
<td>15</td>
<td>0.01848792173</td>
<td>0.00386589960</td>
<td>7.65807822810896 $\times 10^{-4}$</td>
<td>7.65807822810896 $\times 10^{-4}$</td>
</tr>
<tr>
<td>16</td>
<td>0.02710185325</td>
<td>0.00394153801</td>
<td>7.72849859123260 $\times 10^{-4}$</td>
<td>9.1238589363973 $\times 10^{-4}$</td>
</tr>
<tr>
<td>17</td>
<td>0.01733093478</td>
<td>0.00315919427</td>
<td>8.18845196308535 $\times 10^{-4}$</td>
<td>8.18845196308535 $\times 10^{-4}$</td>
</tr>
<tr>
<td>18</td>
<td>0.15382974090</td>
<td>0.00911291945</td>
<td>9.78920394368993 $\times 10^{-4}$</td>
<td>9.74877116643424 $\times 10^{-4}$</td>
</tr>
<tr>
<td>19</td>
<td>0.03034968922</td>
<td>0.00516568726</td>
<td>9.6160668146433 $\times 10^{-4}$</td>
<td>9.61606681462893 $\times 10^{-4}$</td>
</tr>
<tr>
<td>20</td>
<td>0.02073031734</td>
<td>0.00363290423</td>
<td>7.8974053695927 $\times 10^{-4}$</td>
<td>7.8974053695927 $\times 10^{-4}$</td>
</tr>
</tbody>
</table>

Average | 0.03536683382 | 0.00443603175 | 8.7045934382246 $\times 10^{-4}$ | 8.98786441303749 $\times 10^{-4}$ |
coefficients are investigated and solved. The numerical-testing results demonstrate the efficacy and superiority of the presented E47 algorithm for QP solving. In addition, to further demonstrate the advantages, the comparison results between the E47 algorithm and the active set algorithm are provided, which illustrate the high accuracy and the (168 times) faster convergence of the E47 algorithm. The numerical results and discussions provide interesting insights and new answers to related quadratic programming problems.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under Grants 61075121 and 60935001, and also by the Fundamental Research Funds for the Central Universities of China. In addition, the corresponding author, Y. Zhang, would like to thank and encourage the student coauthors by sharing the following thoughts: “Researchers have to notice and handle the contradiction between the two basic properties of science (or to say, scientific research), i.e., creativity and repeatability. So, the distance between two incremental-research papers should always be maximized (i.e., as large as possible), which, however, should also be appropriately small (i.e., as small as acceptable) so that the authors and readers can figure out the results of the second paper from those of the first one”.

REFERENCES


<table>
<thead>
<tr>
<th>Test #</th>
<th>Time\text{active _ set} (s)</th>
<th>Time\text{E47} (s)</th>
<th>Solutions' difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.876790495635962</td>
<td>0.02810104713820</td>
<td>5.36625842423164×10^{-4}</td>
</tr>
<tr>
<td>2</td>
<td>0.860737523922327</td>
<td>0.01795118476933</td>
<td>2.24223137788206×10^{-4}</td>
</tr>
<tr>
<td>3</td>
<td>0.863932241674177</td>
<td>0.03199583371306</td>
<td>3.52456985626206×10^{-4}</td>
</tr>
<tr>
<td>4</td>
<td>0.867815193755354</td>
<td>0.04852982729496</td>
<td>4.10864301641495×10^{-4}</td>
</tr>
<tr>
<td>5</td>
<td>0.873233059165588</td>
<td>0.01848792173231</td>
<td>1.77530795491561×10^{-4}</td>
</tr>
<tr>
<td>6</td>
<td>0.857095121028744</td>
<td>0.02710185325268</td>
<td>2.45043982068672×10^{-4}</td>
</tr>
<tr>
<td>7</td>
<td>0.860923797914806</td>
<td>0.01730394738300</td>
<td>1.73185740865999×10^{-4}</td>
</tr>
<tr>
<td>8</td>
<td>0.565073070648707</td>
<td>0.15382974090059</td>
<td>0.00159630234733×10^{-0}</td>
</tr>
<tr>
<td>9</td>
<td>0.870159758953872</td>
<td>0.03304986291963</td>
<td>3.97930130362768×10^{-4}</td>
</tr>
<tr>
<td>10</td>
<td>0.859110024971292</td>
<td>0.02073031733509</td>
<td>2.39698227434859×10^{-4}</td>
</tr>
</tbody>
</table>

Average: 0.835487073847584 × 0.03971080513935 = 4.353863522012926×10^{-4}