Symbolic-interval heuristic for bound-constrained minimization*

Evgueni Petrov

IRIN — Université de Nantes
2 rue de la Houssinière BP 92208 44322 Nantes Cedex 03 France
evgueni.petrov@irin.univ-nantes.fr

Abstract. Global optimization subject to bound constraints helps answer many practical questions in chemistry, molecular biology, economics. Most of algorithms for solution of global optimization problems are a combination of interval methods and exhuastive search. The efficiency of such algorithms is characterized by their ability to detect and eliminate sub-optimal feasible regions. This ability is increased by availability of a good upper bound on the global minimum. In this paper, we present a symbolic-interval algorithm for calculation of upper bounds in bound-constrained global minimization problems and report the results of some experiments.

1 Introduction

Global optimization subject to bound constraints helps answer many practical questions in chemistry, molecular biology, economics. Most of algorithms for solution of global optimization problems are a combination of interval methods and exhuastive search [2, 4, 9]. The efficiency of such algorithms is characterized by their ability to detect and eliminate sub-optimal feasible regions. This ability is increased by availability of a good upper bound on the global minimum.

A non-trivial upper bound on the minimum of any rational function can be calculated using modal intervals or Kaucher arithmetic [7, 8]. As far as non-linear functions are concerned in general, there is no similar means at present and one has to use some ad hoc combination of sampling and local search. The advantage of this approach is that it is applicable to any function specified by a black box transforming arguments into values. Its most significant disadvantage is that it ignores the symbolic representation of the minimized function which carries the helpful information on the global behaviour of the function.

In the this paper, we present a symbolic-interval algorithm for calculation of upper bounds in bound-constrained global minimization problems and report the results of some experiments.

The paper is structured as follows. Section 2 introduces the basic definitions. Section 3 presents the algorithm. In Section 4 we report the data of experiments with the Dixon-Szegö and some other classical functions. Section 5 concludes the paper. Appendix A contains the test functions.

* Financially supported by Centre Franco-Russe Liapunov (Project 06–98), by European project COCONUT IST–2000–26063.

2 Basic definitions, notation

This section contains the definitions linking real and interval functions, their symbolic representation and partial derivatives.

Intervals are closed convex sets of real numbers, boxes are interval vectors. The set of real numbers and the set of intervals are denoted by $\mathbb{R}$, $\mathbb{I}$. Interval function $[f]$ which returns the exact interval bound on the values of real function f given interval bounds on each of its real arguments is called *interval extension* of f. We call addition, subtraction, multiplication, division, power function for integer exponent, sine, cosine, tangent, exponent function and their inverses *basic functions*. We write "vector", "function" instead of "real vector", "real function".

The symbols $\mathcal{B}$ which denote the basic functions are called *function symbols*. The symbols of the arithmetic operations are *binary*, the others are *unary*. The symbols which denote real numbers are called *constant symbols*. The symbols v_i's distinct from the constant and function symbols are called *variables*.

The set of *terms* over the variables v_i's built from the symbols $\mathcal{B}$ and the constant symbols is defined as usual. A term t' is a *subterm* of a term t, or $t' \sqsubseteq t$ in symbols, if t' occurs in t. The term built from a symbol $\alpha \in \mathcal{B}$ and terms t', t'' is written as $\alpha(t', t'')$ (for binary α) or $\alpha(t')$ (for unary α). The set of variables in t is denoted by $\mathrm{VAR}(t)$.

Consider the following algebras having the same function symbols $\alpha \in \mathcal{B}$: terms $\mathcal{T}_n$ over the variables $v_1, \ldots, v_n$ where (the operation denoted by) α returns the term built from the arguments (of α) and symbol α; functions $\mathcal{R}_n$ of n-dimensional vectors where α returns the composition of the arguments and the basic function denoted by α; interval functions $\mathcal{I}_n$ of n-dimensional boxes where α returns the composition of the arguments and the interval extension of the basic function denoted by α.

Algebra $\mathcal{T}_n$ is called the algebra of *real terms*. The subalgebra of $\mathcal{R}_n$ generated by the projections from $\mathbb{R}^n$ to $\mathbb{R}$ and the constant functions is called the algebra of *elementary functions*. The subalgebra of $\mathcal{I}_n$ generated by the projections from $\mathbb{I}^n$ to $\mathbb{I}$ and the functions returning singleton intervals is called the algebra of *interval elementary functions*.

Elementary and interval elementary functions are images of real terms under the homomorphisms $(\cdot)^{\mathcal{R}_n}$ and $[\cdot]_n$ which map the variables and the constant symbols to the generators of the respective algebra. We say that a term t *specifies* the elementary function $t^{\mathcal{R}_n}$ and interval elementary function $[t]_n$. In what follows, n is some fixed non-negative integer; we write $(\cdot)^{\mathcal{R}}$, $[\cdot]$ instead of $(\cdot)^{\mathcal{R}_n}$, $[\cdot]_n$.

The term α' specifies the derivative of the function specified by $\alpha(v_1)$ where symbol $\alpha \in \mathcal{B}$ is unary. The terms α'_1, α'_2 specify the partial derivatives of the function specified by $\alpha(v_1, v_2)$ where α is binary. The term obtained from some term t by simultaneous substitution of terms t_i's for the variables v_i's is denoted by $t(t_1, \ldots, t_n)$. In the case where $\mathrm{VAR}(t)$ is $\{v_1, v_2\}$ or $\{v_1\}$, we use the abbreviations $t(t_1, t_2)$, $t(t_1)$.

The function $\partial t/\partial(\cdot)$ recursively defined on the subterms of term t as follows:

$$\begin{aligned}
\partial t/\partial t &= 1 \\
\partial t/\partial t' &= \alpha'(t') \cdot \partial t/\partial \alpha(t') && \text{for some unary } \alpha \in \mathcal{B}, \text{ some } \alpha(t') \sqsubseteq t \\
\partial t/\partial t' &= \alpha'_1(t', t'') \cdot \partial t/\partial \alpha(t', t'') && \text{for some binary } \alpha \in \mathcal{B}, \text{ some } \alpha(t', t'') \sqsubseteq t \\
\partial t/\partial t' &= \alpha'_2(t'', t') \cdot \partial t/\partial \alpha(t'', t') && \text{for some binary } \alpha \in \mathcal{B}, \text{ some } \alpha(t'', t') \sqsubseteq t
\end{aligned}$$

is called *differentiation* of t. The term $\partial t/\partial t'$ is called *partial derivative* of t wrt $t' \sqsubseteq t$. We assume that the symbols $\cdot$ and 1 denote multiplication and the unit.

Suppose that t contains a single occurence of t'. Let $t = t''(v_1, \ldots, v_n, t')$ where $\text{VAR}(t'') = \{v_1, \ldots, v_{n+1}\}$. The function $(\partial t/\partial t')^{\mathcal{R}} : \mathbb{R}^n \to \mathbb{R}$ is the composition of the partial derivative $D_{n+1}\left[(t'')^{\mathcal{R}_{n+1}}\right]$ wrt the last argument of the function $(t'')^{\mathcal{R}_{n+1}} : \mathbb{R}^{n+1} \to \mathbb{R}$ and the injection from $\mathbb{R}^n$ to $\mathbb{R}^{n+1}$ which sends every vector p to the vector $\big(p, (t')^{\mathcal{R}}(p)\big)$.

3 Heuristic for bound-constrained minimization

The global minimum of any function can be bounded from above with the help of sampling and local search. Applying local search to a function that is non-convex a priori has two drawbacks: the risk of slow convergence and the risk of violation of bound constraints. These risks decrease as the point for starting local search approaches to the global minimizer. The goal for our algorithm is to find a point that satisfies the bound constraints and where the value of the objective is sufficiently small.

Outline of the algorithm Let real term t, box b specify the objective, the bound constraints. Our algorithm repeatedly exploits the following observation concerning the minimum of the objective $t^{\mathcal{R}}$ in b. Let $t^{\mathcal{R}}$ satisfy $\forall p \in \mathbb{R}^n \;\; t^{\mathcal{R}}(p) = f(p, g(p))$ for some elementary functions f and g, and let a be the value of g at some point in b. Then the minimum of the objective is bounded from above by $\min_{p \in b \cap g^{-1}(a)} f(p, a)$ where $g^{-1}(a)$ is the pre-image of a under g.

The choice of a specific a is determined by the considerations of efficiency. The ideal (impractical) choice would be the value of g at the global minimizer of the objective. In practice, we demand that f be monotonic in the last argument and that mini- and maximization of g be affordable. Under these assumptions, we use $a = \min_b g$ (if f increases) or $a = \max_b g$ (if f decreases). The assumptions are valid, if g is specified by a subterm $t' \sqsubseteq t$ such that (1) no variable occurs in t' twice and (2) the function specified by the partial derivative $\partial t/\partial t'$ does not change sign in b. The maximal wrt $\sqsubseteq$ subterms of t that have these two properties are called *good subterms*. Our algorithm is provided in Fig. 1 and Fig. 2.

Example Let us trace the algorithm for the Branin function specified by

$$t = \left(v_2 - \frac{5.1}{4\pi^2}v_1^2 + \frac{5v_1}{\pi} - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos v_1 + 10$$

in the box $b = ([-5, 10], [0, 15])$ (we write t in the infix notation).

There is only one good subterm $t' = 10\left(1 - \frac{1}{8\pi}\right)\cos v_1 \sqsubseteq t$ (one can check that $\partial t/\partial t' = 1 \cdot 1 \cdot 1$). Minimization of $(t')^{\mathcal{R}}$ in b assigns $-\pi$ to v_1 (or π, or 3π depending on the actual implementation of $\text{PROJ}(\cdot, \cdot)$). Since $\text{VAR}(t) = \{v_2\} \neq \emptyset$, the body of the loop is run once again. Now the entire term t is its own good subterm (v_1 has been replaced with $-\pi$!). Minimization of $t^{\mathcal{R}}$ assigns 12.275 to v_2 and the loop terminates for $\text{VAR}(t)$ is now $\emptyset$. The point p for starting local search is $(-\pi, 12.275)$. Actually, it is one of the global minimizers of the Branin function in b.

$p = (0, \ldots, 0)$
WHILE VAR$(t) \neq \emptyset$
  $\ell =$ THE SET OF GOOD SUBTERMS OF t
  IF $\ell = \emptyset$
    BREAK
  IF VAR$(t') =$ VAR(t) FOR EACH $t' \in \ell$
    SET t' TO SUCH SUBTERM FROM ℓ THAT
    $t^{\mathcal{R}}(\text{PROJ}(t', \text{mid}[t'](b) - \text{sign}(\partial t/\partial t') \cdot \text{rad}[t'](b)))$ IS MINIMUM
  ELSE IF VAR(t') IS A SINGLETON SET FOR SOME $t' \in \ell$
    SET t' TO SUCH SUBTERM
  ELSE
    SET t' TO SOME SUBTERM FROM ℓ
  $p = p +$ PROJ$(t', \text{mid}[t'](b) - \text{sign}(\partial t/\partial t') \cdot \text{rad}[t'](b))$
  SUBSTITUTE p_i FOR EACH $v_i \in$ VAR(t') IN t
SET p_i TO $\text{mid}(b_i)$ FOR EACH $v_i \in$ VAR(t)

Fig. 1. The heuristic for bound-constrained minimization; vector p is the point for starting local search, set ℓ is the set of good subterms of t, term $t' \sqsubseteq t$ is some good subterm, function PROJ$(\cdot, \cdot)$ is defined in Fig. 2, function sign$(\cdot)$ takes a term specifying a function that does not change sign in b and returns its sign ($+1$ or -1), functions mid$(\cdot)$ and rad$(\cdot)$ return the midpoint and the radius of intervals, interval b_i is the i-th element in b

PROJ$(v_i, c) = (0, \ldots, 0, c, 0, \ldots, 0)$	where c takes the i-th place
PROJ$(\alpha(t'), c) =$ PROJ(t', c')	where $c = \alpha^{\mathcal{R}}(c')$, $c' \in [t'](b)$
PROJ$(\alpha(t', t''), c) =$ PROJ$(t', c') +$ PROJ(t'', c'')	where $c = \alpha^{\mathcal{R}}(c', c'')$, $c' \in [t'](b)$, $c'' \in [t''](b)$

Fig. 2. The definition of the function PROJ$(\cdot, \cdot)$; symbol v_i is a variable, c, c', c'' are real numbers, terms t', t'' are subterms of t, function $\alpha^{\mathcal{R}}$ is the basic function denoted by $\alpha \in \mathcal{B}$

Correctness The algorithm in Fig. 1 and Fig. 2 is correct; that is it generates a vector p that satisfies each bound constraint.

Correctness of the algorithm follows from the fact that, for the terms t' where each variable occurs no more than once, the interval function $[t']$ is the interval extension of the function $(t')^{\mathcal{R}}$. In particular, it means that the bounds of $[t'](b)$ are the global extrema of $(t')^{\mathcal{R}}$ in b. Therefore the i-th element in the vector PROJ(t', a) where a is any of the bounds of $[t'](b)$ belongs to b_i whenever $v_i \in$ VAR(t').

The worst case complexity In the worst case, the number of evaluations of the basic functions and their interval extensions is $O(s^2/n)$, the number of comparisons is $O(n \cdot s)$ where s is the size of the term specifying the objective, n is the number of the variables.

Computer implementation details The computer implementation of the algorithm in Fig. 1 and Fig. 2 must operate correctly on the real numbers non-representable in the floating point form. Instead of such real numbers, we use thin intervals bounded by

floating point numbers. That is the computer implementation of our algorithm needs 4 floating numbers per interval.

The intervals $[\partial t/\partial t'](b)$ and $[t'](b)$ are calculated for all subterms $t' \sqsubseteq t$ by the interval version of automatic differentiation [3].

4 Experiments with the Dixon-Szegö and some other functions

In this section, we summarize the experiments with a C++ implementation of the algorithm described in Section 3. The functions for these experiments are taken from [1], [6] and the 1st International Contest on Evolutionary Optimization (see Appendix A).

For each test function, we report the number n of the variables in the corresponding term, the global minimum and the upper bound calculated by our heuristic (the 4 first digits from their decimal representation) (see Fig. 3). The general observation concerning the algorithm is that it tends to perform better on separable objective functions.

Dixon-Szegö functions			
function	n	min	upper b.
Shekel 5	4	–10.38	–10.38
Shekel 7	4	–10.48	–10.48
Shekel 10	4	–10.53	–10.53
Goldstein-Price	2	3.000	600.0
6 hump camel	2	–1.031	150.9
Hartmann 3	3	–3.862	–3.761
Hartmann 6	6	–3.322	–3.203
Shubert	2	–186.7	19.87
Branin	2	0.3978	0.3978
Janson-Knüppel functions			
Rosenbrock	2 through 80	0.000	0.000
Levy 8	2 through 80	0.000	0.000

Janson-Knüppel functions (cntd.)			
function	n	min	upper b.
Levy 8 bis	2 through 80	0.000	0.000
Levy 13	2 through 80	0.000	0.000
Int. Contest on Evol. Optim. functions			
ICEO 2	5	0.000	0.000
ICEO 2	10	0.000	0.000
ICEO 3	5	–10.40	–10.40
ICEO 3	10	–10.20	–10.20
ICEO 4	5	–4.687	–4.488
ICEO 4	10	–9.660	–8.048
ICEO 5	5	–1.499	–1.499
ICEO 5	10	–1.500	–1.500

Fig. 3. The upper bounds found by the algorithm for the Dixon-Szegö, Janson-Knüppel and 1st International Constest on Evolutionary Optimization functions; each line contains the name of a function from Appendix A, the dimension(s) tried for this function, its global minimum, the calculated upper bound

5 Conclusion

In this paper we have presented a symbolic-interval heuristic for bound-constrained minimization problems. The key idea of our heuristic is to simplify the minimization problem by ignoring the dependencies between certain subexpressions in the symbolic representation of the objective. Our experiments indicate that this approach allows one to calculate good upper bounds in a number of bound-constrained minimization problems.

Our future work will focus on the theoretical properties of the presented heuristic as well as on its integration into modern algorithms for global optimization like [5].

References

1. L. C. W. Dixon and G. P. Szego, editors. *Towards global Optimization*. North-Holland, Amsterdam, 1975–1978.
2. A. V. Fiacco and G. P. McCormick. *Nonlinear Programming. Sequential unconstrained minimization techniques*. Willey, reprinted by SIAM publications, 1990.
3. A. Griewank. On automatic differentiation. In M. Iri and K. Tanabe, editors, *Mathematical Programming: Recent Developments and Applications*, pages 83–108. Kluwer Academic Publishers, 1989.
4. E. Hansen. *Global optimization using interval analysis*, volume 165 of *Pure and applied mathematics*. Marcel Dekker, 1992.
5. W. Huyer and A. Neumaier. Global optimization by multilevel coordinate search. *Journal of Global Optimization*, 14:331–355, 1999.
6. C. Jansson and O. Knüppel. A global minimization method: the multi-dimensional case. Technische Informatik-III, TU Hamburg-Harburg, 1992.
7. E. Kaucher. Interval analysis in the extended interval space IR. *Computing*, Suppl. 2:33–49, 1980.
8. A. Llobet. Application of modal interval analysis to the simulation of the behaviour of dynamic systems with uncertain parameters. Ph. D. thesis, 1999.
9. J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, 1999.

A Test functions

For the sake of completeness we give the terms and the boxes that specify the test objective functions and the corresponding bound constraints. We denote the n-element vector $(x, \ldots, x)$ by x^n. The terms are written in the infix notation. The symbolic sums and products must be expanded into the appropriate chains of left-associative additions and multiplications.

Shekel m $b = [0, 10]^4$, $t = -\sum_{i=1}^{m} 1 \Big/ \left(c_i + \sum_{j=1}^{4} (v_j - a_{ij})^2\right)$

$$c = \begin{pmatrix} 1 & 2 & 2 & 4 & 4 & 6 & 3 & 7 & 5 & 5 \end{pmatrix} / 10, \quad a^{\mathrm{T}} = \begin{pmatrix} 4 & 1 & 8 & 6 & 3 & 2 & 5 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 5 & 1 & 2 & 3.6 \\ 4 & 1 & 8 & 6 & 3 & 2 & 3 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 3 & 1 & 2 & 3.6 \end{pmatrix}$$

Goldstein–Price $b = [-2, 2]^2$,

$$t = \left(1 + (v_1 + v_2 + 1)^2 \cdot \left(19 - 14 \cdot v_1 + 3 \cdot v_1^2 - 14 \cdot v_2 + 6 \cdot v_1 \cdot v_2 + 3 \cdot v_2^2\right)\right) \times \left(30 + (2 \cdot v_1 - 3 \cdot v_2)^2 \cdot \left(18 - 32 \cdot v_1 + 12 \cdot v_1^2 + 48 \cdot v_2 - 36 \cdot v_1 \cdot v_2 + 27 \cdot v_2^2\right)\right)$$

6 hump camel

$$b = ([-3, 3], [-2, 2]), \quad t = \left(4 - 2.1 \cdot v_1^2 + v_1^4/3\right) \cdot v_1^2 + v_1 \cdot v_2 + 4 \cdot \left(v_2^2 - 1\right) \cdot v_2^2$$

Hartmann 3 $b = [0,1]^3$, $t = -\sum_{j=1}^{4} c_j \cdot \exp\left(-\sum_{i=1}^{3} a_{ij} \cdot (v_i - p_{ij})^2\right)$,

$$a = \begin{pmatrix} 3 & 0.1 & 3 & 0.1 \\ 10 & 10 & 10 & 10 \\ 30 & 35 & 30 & 35 \end{pmatrix}, \quad p = \begin{pmatrix} 0.36890 & 0.46990 & 0.10910 & 0.03815 \\ 0.11700 & 0.43870 & 0.87320 & 0.57430 \\ 0.26730 & 0.74700 & 0.55470 & 0.88280 \end{pmatrix},$$
$$c = \begin{pmatrix} 1 & 1.2 & 3 & 3.2 \end{pmatrix}$$

Hartmann 6 $b = [0,1]^6$, $t = -\sum_{j=1}^{4} c_j \cdot \exp\left(-\sum_{i=1}^{6} a_{ij} \cdot (v_i - p_{ij})^2\right)$,

$$a^{\mathrm{T}} = \begin{pmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{pmatrix}, p^{\mathrm{T}} = \begin{pmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{pmatrix},$$
$$c = \begin{pmatrix} 1 & 1.2 & 3 & 3.2 \end{pmatrix}$$

Shubert $b = [-10,10]^2$, $t = \prod_{j=1}^{2} \sum_{i=1}^{6} i \cdot \cos((i+1) \cdot v_j + i)$

Rosenbrock $b = [-2.048, 2.048]^n$, $t = \sum_{i=1}^{n-1} 100 \cdot \left(v_i - v_{i+1}^2\right)^2 + (1 - v_{i+1})^2$

Levy 8 $b = [-10,10]^n$,

$$t = 10 \cdot \sin^2 \pi \cdot \frac{v_1 + 3}{4} + \left(\frac{v_n - 1}{4}\right)^2 + \sum_{i=1}^{n-1} \left(\frac{v_i - 1}{4}\right)^2 \cdot \left(1 + 10 \cdot \sin^2 \pi \cdot \frac{v_{i+1} + 3}{4}\right)$$

Levy 8 bis

$$b = [-10,10]^n, \quad t = 10 \cdot \sin^2 \pi \cdot v_1 + (v_n - 1)^2 + \sum_{i=1}^{n-1} (v_i - 1)^2 \cdot \left(1 + 10 \cdot \sin^2 \pi \cdot v_{i+1}\right)$$

Levy 13 $b = [-10,10]^n$,

$$t = \sin^2 3 \cdot \pi \cdot v_1 + (v_n - 1)^2 \cdot \left(1 + \sin^2 3 \cdot \pi \cdot v_n\right) + \sum_{i=1}^{n-1} (v_i - 1)^2 \cdot \left(1 + \sin^2 3 \cdot \pi \cdot v_{i+1}\right)$$

ICEO 2 $b = [-600,600]^n$, $t = \sum_{i=1}^{n} (v_i - 100)^2 / 4000 - \prod_{i=1}^{n} \cos\left((v_i - 100) \Big/ \sqrt{i}\right) + 1$

ICEO 3 $b = [0,10]^n$, $t = -\sum_{j=1}^{30} 1 \Big/ \left(c_j + \sum_{i=1}^{n} (v_i - a_{ji})^2\right)$ (see the matrix a and the vector c below)

ICEO 4 $b = [0,\pi]^n$, $t = -\sum_{i=1}^{n} \sin v_i \cdot \sin^{20}\left(i \cdot v_i^2 / \pi\right)$

ICEO 5 See the matrix a and the vector c below.

$$b = [0,10]^n, \quad t = -\sum_{j=1}^{30} c_j \cdot \exp\left(-\sum_{i=1}^{n} (a_{ji} - v_i)^2 / \pi\right) \cdot \cos\left(\pi \cdot \sum_{i=1}^{n} (a_{ji} - v_i)^2\right)$$

The coefficients for ICEO 3 and ICEO 5

$$a = \begin{pmatrix}
9.681 & 0.667 & 4.783 & 9.095 & 3.517 & 9.325 & 6.544 & 0.211 & 5.122 & 2.020 \\
9.400 & 2.041 & 3.788 & 7.931 & 2.882 & 2.672 & 3.568 & 1.284 & 7.033 & 7.374 \\
8.025 & 9.152 & 5.114 & 7.621 & 4.564 & 4.711 & 2.996 & 6.126 & 0.734 & 4.982 \\
2.196 & 0.415 & 5.649 & 6.979 & 9.510 & 9.166 & 6.304 & 6.054 & 9.377 & 1.426 \\
8.074 & 8.777 & 3.467 & 1.863 & 6.708 & 6.349 & 4.534 & 0.276 & 7.633 & 1.567 \\
7.650 & 5.658 & 0.720 & 2.764 & 3.278 & 5.283 & 7.474 & 6.274 & 1.409 & 8.208 \\
1.256 & 3.605 & 8.623 & 6.905 & 0.584 & 8.133 & 6.071 & 6.888 & 4.187 & 5.448 \\
8.314 & 2.261 & 4.224 & 1.781 & 4.124 & 0.932 & 8.129 & 8.658 & 1.208 & 5.762 \\
0.226 & 8.858 & 1.420 & 0.945 & 1.622 & 4.698 & 6.228 & 9.096 & 0.972 & 7.637 \\
7.305 & 2.228 & 1.242 & 5.928 & 9.133 & 1.826 & 4.060 & 5.204 & 8.713 & 8.247 \\
0.652 & 7.027 & 0.508 & 4.876 & 8.807 & 4.632 & 5.808 & 6.937 & 3.291 & 7.016 \\
2.699 & 3.516 & 5.874 & 4.119 & 4.461 & 7.496 & 8.817 & 0.690 & 6.593 & 9.789 \\
8.327 & 3.897 & 2.017 & 9.570 & 9.825 & 1.150 & 1.395 & 3.885 & 6.354 & 0.109 \\
2.132 & 7.006 & 7.136 & 2.641 & 1.882 & 5.943 & 7.273 & 7.691 & 2.880 & 0.564 \\
4.707 & 5.579 & 4.080 & 0.581 & 9.698 & 8.542 & 8.077 & 8.515 & 9.231 & 4.670 \\
8.304 & 7.559 & 8.567 & 0.322 & 7.128 & 8.392 & 1.472 & 8.524 & 2.277 & 7.826 \\
8.632 & 4.409 & 4.832 & 5.768 & 7.050 & 6.715 & 1.711 & 4.323 & 4.405 & 4.591 \\
4.887 & 9.112 & 0.170 & 8.967 & 9.693 & 9.867 & 7.508 & 7.770 & 8.382 & 6.740 \\
2.440 & 6.686 & 4.299 & 1.007 & 7.008 & 1.427 & 9.398 & 8.480 & 9.950 & 1.675 \\
6.306 & 8.583 & 6.084 & 1.138 & 4.350 & 3.134 & 7.853 & 6.061 & 7.457 & 2.258 \\
0.652 & 2.343 & 1.370 & 0.821 & 1.310 & 1.063 & 0.689 & 8.819 & 8.833 & 9.070 \\
5.558 & 1.272 & 5.756 & 9.857 & 2.279 & 2.764 & 1.284 & 1.677 & 1.244 & 1.234 \\
3.352 & 7.549 & 9.817 & 9.437 & 8.687 & 4.167 & 2.570 & 6.540 & 0.228 & 0.027 \\
8.798 & 0.880 & 2.370 & 0.168 & 1.701 & 3.680 & 1.231 & 2.390 & 2.499 & 0.064 \\
1.460 & 8.057 & 1.336 & 7.217 & 7.914 & 3.615 & 9.981 & 9.198 & 5.292 & 1.224 \\
0.432 & 8.645 & 8.774 & 0.249 & 8.081 & 7.461 & 4.416 & 0.652 & 4.002 & 4.644 \\
0.679 & 2.800 & 5.523 & 3.049 & 2.968 & 7.225 & 6.730 & 4.199 & 9.614 & 9.229 \\
4.263 & 1.074 & 7.286 & 5.599 & 8.291 & 5.200 & 9.214 & 8.272 & 4.398 & 4.506 \\
9.496 & 4.830 & 3.150 & 8.270 & 5.079 & 1.231 & 5.731 & 9.494 & 1.883 & 9.732 \\
4.138 & 2.562 & 2.532 & 9.661 & 5.611 & 5.500 & 6.886 & 2.341 & 9.699 & 6.500
\end{pmatrix}$$

$$c = \begin{pmatrix}
0.806 \\
0.517 \\
0.1 \\
0.908 \\
0.965 \\
0.669 \\
0.524 \\
0.902 \\
0.531 \\
0.876 \\
0.462 \\
0.491 \\
0.463 \\
0.714 \\
0.352 \\
0.869 \\
0.813 \\
0.811 \\
0.828 \\
0.964 \\
0.789 \\
0.360 \\
0.369 \\
0.992 \\
0.332 \\
0.817 \\
0.632 \\
0.883 \\
0.608 \\
0.326
\end{pmatrix}$$