אורט בראודה
המכללה האקדמית להנדסה
ORT Braude College

המחלקה למתמטיקה
Department of Mathematics

פרויקט מסכם לתואר בוגר במדעים (B.Sc) במתמטיקה שימושית

שיטת דאגלס-רצפורד לפתרון בעיות היתכנות קמורות ולא קמורות

מוחמד דלאשה

The Douglas-Rachford method for solving convex and non-convex feasibility problems

Mohammed Dallasheh

פרויקט מסכם לתואר בוגר במדעים (B.Sc)
במתמטיקה שימושית

שיטת דאגלס-רצפורד לפתרון בעיות
היתכנות קמורות ולא קמורות

מוחמד דלאשה

The Douglas-Rachford method
for solving convex and
non-convex feasibility problems

Mohammed Dallasheh

Advisor:
Dr. Aviv Gibali

מנחה:
ד"ר אביב גיבלי

Karmiel

כרמיאל

2019

המכללה האקדמית להנדסה אורט בראודה (ע"ר) רח' סנונית 51 ת.ד. 78 כרמיאל, 21982 טל' 04-9901911 פקס 04-9882016
Ort Braude College, P.O.Box 78, Karmiel 21982, Israel, Tel: 972-4-99901911, Fax: 972-4-9882016

Contents

1 Introduction **4**

2 Preliminaries **5**

2.1 The classical Douglas–Rachford method 8

3 Implementation **10**

3.1 Convex setting 10

3.2 Non-convex setting 12

3.2.1 The eight queen problem 12

3.2.2 Sudoku 18

4 Conclusions **26**

List of Figures

1 Convex and non-convex sets . . . 5
2 (Projection). Point B is the projection of A onto the circle. . . . 6
3 The iterative step of the Douglas–Rachford algorithm with the two-set A and B. . . . 8
4 Solid black arrows represent 2-set Douglas–Rachford iterations. . . . 10
5 Three circles example. . . . 11
6 Five circles example. . . . 12
7 The average number iterations needed for solving a 10 balls CFP in varying dimensions. . . . 12
8 Standard queen's move. . . . 13
9 A Chess board 13
10 Row constraints. . . . 14
11 Column constraints. . . . 15
12 Positive slope diagonal. . . . 15
13 Positive slope diagonal. . . . 16
14 Projection onto C1. . . . 16
15 Solved in 154 iterations. . . . 17
16 Solved in 117 iterations. . . . 17
17 Solved in 188 iterations. . . . 18
18 Solved in 48 iterations. . . . 18
19 A 9x9 Sudoku. . . . 19
20 Representation of the above as a cubic Sudoku. . . . 19
21 Row constraints. . . . 20
22 Column constraints. . . . 21
23 3×3 constraints. . . . 21
24 Height constraints. . . . 22
25 Solved in 974 iterations. . . . 23
26 Solved in 1915 iterations. . . . 23
27 Solved in 686 iterations. . . . 24
28 Solved in 374 iterations. . . . 24
29 Solved in 556 iterations. . . . 25
30 Solved in 829 iterations. . . . 25

Abstract

In this project we are concern with feasibility problem, convex and non-convex in real Hilbert spaces. We study the performances of the well-known Douglas–Rachford algorithm and illustrate how this algorithm which uses reflections with respect to sets, can be applied successfully to solve system of quadratic equations (convex case) and also combinatorial games such as the Eight Queen puzzle and Sodoku.

This project not only shows the practical advantages of the Douglas–Rachford method, but also suggest to remodel different problems as feasibility problems and then apply and kind of projection method for solving it.

I would like to thank Dr. Aviv Gibali for helping defining this interesting research project and guide me along its process. I would also like to thank the Mathematics Department for providing me all the needed mathematical and programming skills as well as unlimited support throughout my studies.

1 Introduction

In this work we are concerned with feasibility problem and to keep it simple at the beginning we focus on the *Convex Feasibility Problem* (CFP). Given nonempty, closed and convex sets $C_i \subseteq \mathbb{R}^d$ where $i = 1, dots, N$. The CFP is formulated as follows.

$$\text{find a point } x^* \in C := \cap_{i=1}^{N} C_i. \tag{1.1}$$

One of the simplest and earliest methods for solving CFPs are projection methods. Originally these methods were used for solving system of linear equations and inequalities in the Euclidean spaces. Later these methods were improved and generalized to solve general CFPs.

These methods are iterative procedures which use projections, of different types, onto sets by taking into account that the projection onto the intersection of the sets is a very hard computational task, while projections onto the individual sets are relatively easier. This is the reason why these methods are applied successfully in many real-world applications, see [4, 5]

One specific methods which uses reflections and gains a lot of interest in recent years is the Douglas–Rachford algorithm [7]. The method was introduced to solve the following instationary heat equation:

$$\begin{cases} \partial_t u = \partial_{xx} u + \partial_{yy} u \\ u(x, y, 0) = f(x, y). \end{cases} \tag{1.2}$$

where u, f are some functions.

Lions and Mercier [9] were the ones who made the major work in this field and adjusted and extended the algorithm successfully for solving CFPs and even more general problems, such as zero of the sum of two maximally monotone operators.

In this project we first introduce several real-world problems that can be remodeled as a feasibility problem (convex and non-convex) and then apply the Douglas–Rachford algorithm for solving it successfully.

2 Preliminaries

We start with several definitions and notions.

Definition 2.1 *Let X be a vector space and let $C \subseteq X$. We say C is* ***convex*** *if*

$$\lambda C + (1-\lambda)C \subseteq C, \quad \forall \lambda \in [0,1]. \tag{2.1}$$

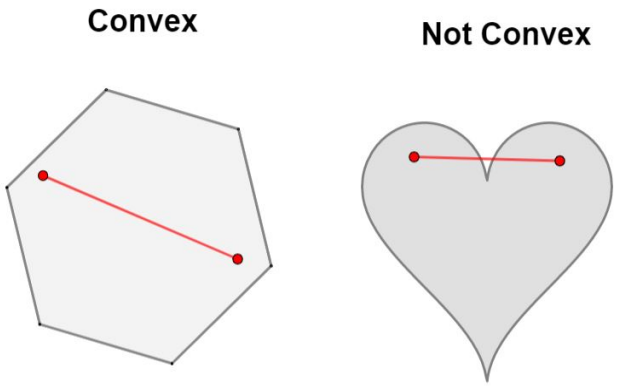


Figure 1: Convex and non-convex sets

See Figure 1 for geometrical illustration of convex and non-convex sets.

Definition 2.2 *Let $C \subset \mathbb{R}^d$ be some closed set.(i) The* ***closest point projection*** *in C is a set-valued mapping $P_C : \mathbb{R}^d \rightrightarrows C$ which assign for any $x \in \mathbb{R}^d$ an element denoted by $P_C(x)$ and is characterized by the fact that $P_C(x) \in C$ and is the solution of the following optimization problem*

$$P_C(x) = \text{Argmin}\{\|z - x\| \mid z \in C\} \tag{2.2}$$

(ii) The ***reflection*** *in the set C is a set-valued mapping $R_C : \mathbb{R}^d \rightrightarrows \mathbb{R}^n$ defined as $R_C(x) = 2P_C(x) - x$.*

See Figure 9 for geometrical illustration.

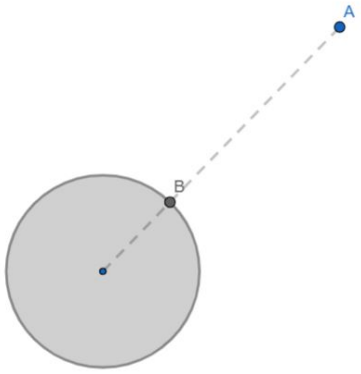


Figure 2: (Projection). Point B is the projection of A onto the circle.

Some useful examples in which there exists close formulas for the projections (hence also the reflections) are given next.

Example 2.3 *Let $C_i \subseteq \mathbb{R}^d$ for $i = 1, \ldots, N$ be closed and convex sets. Consider the* ***product set*** $\mathcal{C} := C_1 \times C_2 \cdots \times C_N \subseteq \mathbb{R}^{Nd}$. *The projection onto $\mathcal{C}$ of a point $x^1, x^2, \ldots, x^N$ ($x^i \in \mathbb{R}^d$) is defined as:*

$$P_{\mathcal{C}}(x^1, x^2, \ldots, x^N) = (P_{C_1}(x^1), P_{C_2}(x^2), \ldots, P_{C_N}(x^N)). \tag{2.3}$$

Proof. Clearly, $P_{C_i}(x_i) \in C_i$. Then $\big(P_{C_1}(x^1), P_{C_2}(x^2), \ldots, P_{C_N}(x^N)\big) \in C_1 \times C_2 \times \cdots \times C_N = \mathcal{C}$. now $\forall c \in \mathcal{C}$, let $c = (c^1, c^2, \ldots, c^N)$ with $c_i \in C_i$. we will show that:

$$\langle c^1 - (P_{C_1}(x^1), \ldots, P_{C_n}(x^N)), (x^1, x^2, \ldots, x^N) - (P_{C_1}(x^1), \ldots, P_{C_N}(x^N))\rangle \leq 0$$

now,

$$\begin{aligned}
&\langle (c^1, \ldots, c^N) - (P_{C_1}(x^1), \ldots, P_{C_N}(x^N)), (x^1, \ldots, x^N) - (P_{C_1}(x^1), \ldots, P_{C_N}(x^N))\rangle \\
&= \langle (c^1 - P_{C_1}(x^1), \ldots, c^N - P_{C_N}(x^N)), (x^1 - P_{C_1}(x^1), \ldots, x^N - P_{C_N}(x^N))\rangle \\
&= \langle c^1 - P_{C_1}(x^2), x^1 - P_{C_1}(x^1)\rangle + \cdots + \langle c^N - P_{C_N}(x^N), x^N - P_{C_N}(x^N)\rangle \leq 0
\end{aligned}$$

because every term above is less than or equal to 0. ■

Example 2.4 *Consider the following set which is called the* `diagonal set`

$$D := \{(x^1, x^2, \ldots, x^N) \in \mathbb{R}^{Nd} \mid x^i \in \mathbb{R}^d\}. \tag{2.4}$$

Given a point $(x^1, x^2, \ldots, x^N) \in \mathbb{R}^{Nd}$*, the projection of it onto* D *is defined as*

$$P_D(x^1, x^2, \ldots, x^N) = (\bar{x}, \bar{x}, \ldots, \bar{x}) \tag{2.5}$$

where $\bar{x} = \frac{1}{N} \sum x_i$

Proof. First observe that D is closed and convex. Clearly,$(\overline{x}, \overline{x}, \ldots, \overline{x}) \in D$, let $c \in D$, say $c = (x, x, \ldots, x)$ for some $x \in \mathbb{R}^d$. Then

$$\begin{aligned}
&\langle c - (\overline{x}, \overline{x}, \ldots, \overline{x}), (x^1, x^2, \ldots, x^N) - (\overline{x}, \overline{x}, \ldots, \overline{x})\rangle \\
&= \langle (x, x, \ldots, x) - (\overline{x}, \ldots, \overline{x}), (x^1, x^2, \ldots, x^N) - (\overline{x}, \ldots, \overline{x})\rangle \\
&= \langle (x - \overline{x}, \ldots, x - \overline{x}), (x^1 - \overline{x}), \ldots, (x^N - \overline{x})\rangle \\
&= \langle x - \overline{x}, x^1 - \overline{x}\rangle + \cdots + \langle x - \overline{x}, x^N - \overline{x}\rangle \\
&= \langle x - \overline{x}, (x^1 - \overline{x}) + \cdots + (x^N - \overline{x})\rangle \\
&= \langle x - \overline{x}, (x^1 + \cdots + x^N) - N\overline{x}\rangle \\
&= \langle x - \overline{x}, N\overline{x} - N\overline{x}\rangle \\
&= \langle x - \overline{x}, 0\rangle \leq 0.
\end{aligned}$$

and the desired result is obtained. ■

Example 2.5 *Denote the standard unit vectors in* $\mathbb{R}^d$ *by* e^i*, and consider the (very) non-convex set of unit vectors* $C = \{e^1, \ldots, e^n\}$*. Then, for a given* $x \in \mathbb{R}^d$ *we have*

$$P_C(x) = \{e^i \mid x_i = max\{x_1, \ldots, x_d\}\}. \tag{2.6}$$

Proof. The set C has a finite number of elements so $P_C(x) \neq \varnothing$. Let $i \in \{1, \ldots, d\}$, then

$$\begin{aligned}
e^i \in P_C(x) &\iff \| x - e^i \| \leq \| x - e^j \| \quad \forall j \\
&\iff \| x - e^i \|^2 \leq \| x - e^j \|^2 \quad j \\
&\iff \| x \|^2 - 2\langle x, e^i\rangle + \| e^i \|^2 \leq \| x \|^2 - 2\langle x, e^j\rangle + \| e^j \|^2 \ \forall j \\
&\iff -2\langle x, e^i\rangle \leq -2\langle x, e^j\rangle \quad \forall j \\
&\iff \langle x, e^j\rangle \leq \langle x, e^j\rangle \quad \forall j \\
&\iff x_j \leq x_i \quad \forall j.
\end{aligned}$$

which completes the proof. ■

2.1 The classical Douglas–Rachford method

In this section we are ready to present and study the Douglas–Rachford method. Given two sets C_1, C_2 of a real Hilbert space $\mathcal{H}$, the Douglas-Rachford operator is defined as:

$$T_{C_1,C_2} = \frac{I + R_{C_2}R_{C_2}}{2}$$

Using the above operator which is known to be nonexpansive, the convergence theorem of the methods is the following.

Theorem 2.6 *Let $C_1, C_2 \subset \mathcal{H}$ be non-empty, closed and convex and let $x^0 \in \mathcal{H}$. Any sequence $\left\{x^{k+1} = T_{C_1,C_2}(x^k)\right\}_{k=0}^{\infty}$ converges weakly to a point z such that $P_{C_1}(z) \in C_1 \cap C_2$.*

See geometrical interpolation of the Douglas–Rachford iterative step for two-sets CFP A, B in Figure 3 (taken Dr. D. Rubén Campoy García thesis [8]).

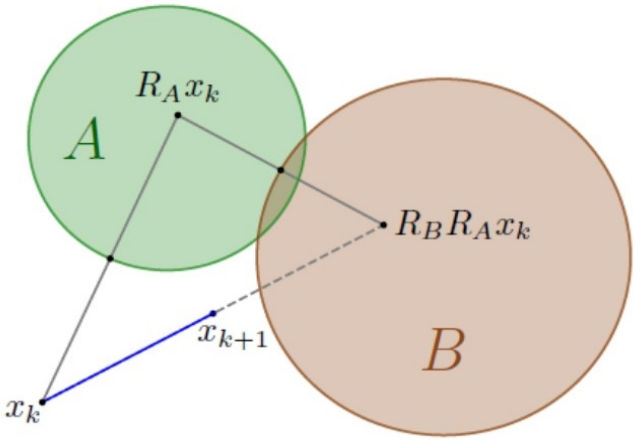


Figure 3: The iterative step of the Douglas–Rachford algorithm with the two-set A and B.

while the classic algorithm can handle just two sets, there exists many techniques to generalize it to solve the general CFP, that is: Find a point x

such that

$$x \in \bigcap_{i=1}^{N} C_i. \tag{2.7}$$

Recall the product and diagonal sets, C, D.

$$\mathcal{C} = C_1 \times C_2 \times, \ldots, \times C_N = \{(c^1, c^2, \ldots, c^N) \mid c^i \in C_i\} \tag{2.8}$$

$$D = \{(x^1, x^2, \ldots, x^N) \mid x^1 = x^2 = \cdots = x^N\} \tag{2.9}$$

Then

$$D \cap \mathcal{C} = \{(c^1, c^2, \ldots, c^N) \mid c^1 = c^2 = \cdots = c^N, c_i \in C_i\}.$$

So based on the above definition we get the following equivalent formulation.

$$x \in \bigcap_{i=1}^{N} C_i \quad \Longleftrightarrow \quad (x, x, \ldots, x) \in \mathcal{C} \cap D$$

Now, the Douglas–Rachford algorithm applied to the sets $\mathcal{C} \cap D$ yields a simultaneous version which operates in the appropriate product space. In recent years several cyclic variants of the Douglas–Rachford method were introduced. For example we present the result of [1, 6, 2, 3].
One cyclic variant of the Douglas–Rachford method requires the following definition. Given

$$T_{[C_1\, C_2 \ldots C_N]} = T_{C_N, C_1}, T_{C_{N-1}, C_N}, \ldots, T_{C_2, C_3}, T_{C_1, C_2}. \tag{2.10}$$

The convergence theorem of the method is given next.

Theorem 2.7 *Let $C_1, C_2, \ldots, C_N \subseteq \mathcal{H}$ be closed and convex sets with a nonempty intersection. For any starting point x_0, define the sequence $x_{k+1} = T_{[C_1\, C_2 \ldots C_N]}(x_k)$. Then $\{x_k\}$ converges weakly to a point x such that $P_{C_i}(x) = P_{C_j}(x)$ for all indices i and j. Moreover, $P_{C_j}(x) \in \bigcap_{i=1}^{N} C_i$*

See Figure 4 for illustration with three lines, taken again from [1].

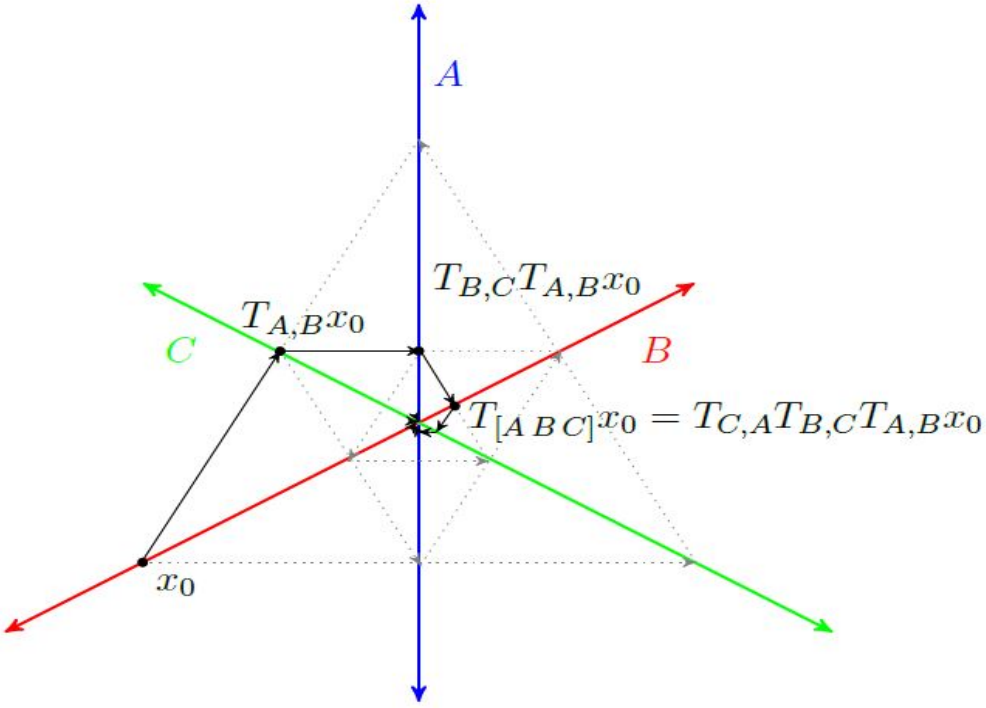


Figure 4: Solid black arrows represent 2-set Douglas–Rachford iterations.

3 Implementation

In this section we present convex and non-convex CFP and apply the Douglas-Rachford method for solving it. When more than two sets are involved, the product space reformulation is used.

3.1 Convex setting

In this subsection we consider a quadratic CFP where the sets C_i are balls in $\mathbb{R}^d$, that is

$$C_i = \{x \in \mathbb{R}^d \mid \|x - x_i\| \leq r_i^2\} \tag{3.1}$$

where x_i is the center of the ball and r_i is the radius. The projection (also the refelction) onto each ball has a closed form:

$$P_{C_i}(x) = \frac{x_i - x}{max\{\|x_i - x\|, 1\}} r_i^2 \tag{3.2}$$

For the numerical testings we generated M random balls in $\mathbb{R}^d$ of various sizes. The idea is to first generate a random point $x^* \in \mathbb{R}^d$ which is the solution of the generated CFP. Then, we pick random centers $x_i \in \mathbb{R}^d$, measure it distance to x^*, increase it by some random number and this is the radius of the i-th ball. This construction ensure that the CFP has a solution x^*. Bellow in Figures 3.1-3.1 trajectories of the DR algorithm in $\mathbb{R}^2$ are given.

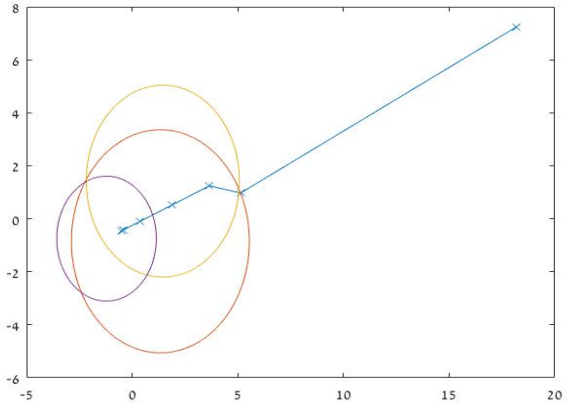


Figure 5: Three circles example.

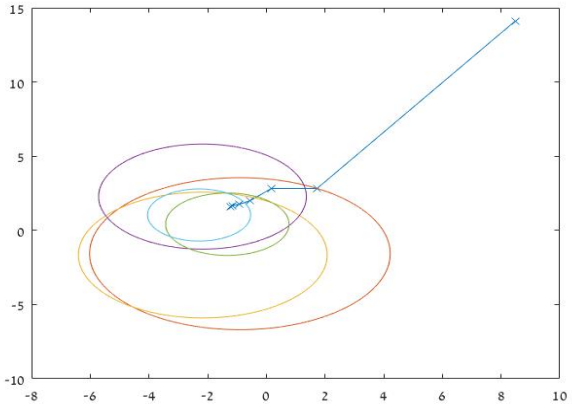


Figure 6: Five circles example.

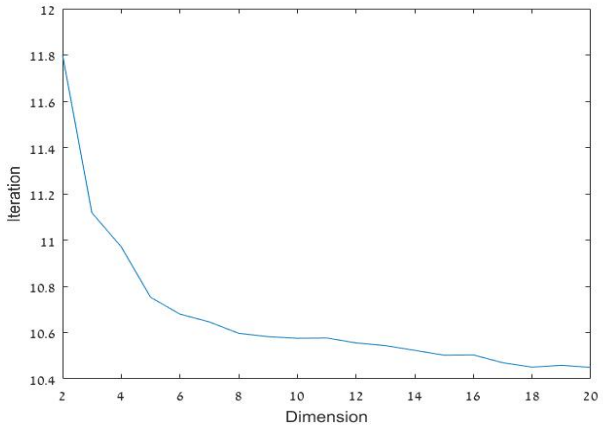


Figure 7: The average number iterations needed for solving a 10 balls CFP in varying dimensions.

3.2 Non-convex setting

3.2.1 The eight queen problem

The 8-queen puzzle is the problem of placing eight queens on a chess board so that any queen threat the others using the standard chess queen move,

this mean that each row, column and diagonal have at most one queen, see Figure 8.

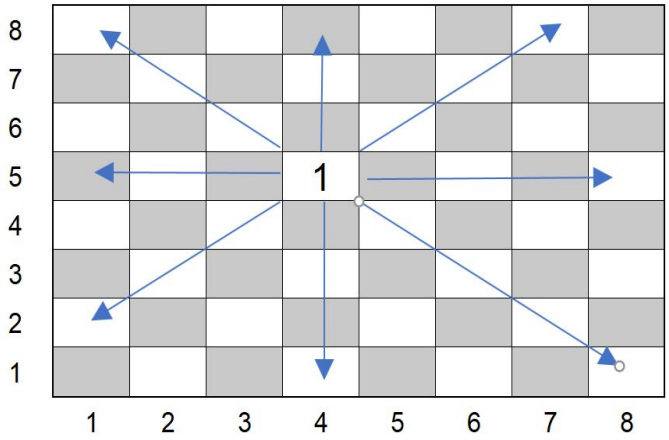


Figure 8: Standard queen's move.

We start by showing how this problem can be remodeled as a feasibility problem. We focus on a 8×8 chess board but clearly this can be extended to any size board. So, consider a 8×8 matrix and place the value 1 if there is a queen in this place on the board and 0 otherwise. See an example in Figure 9. The chess board will look like :

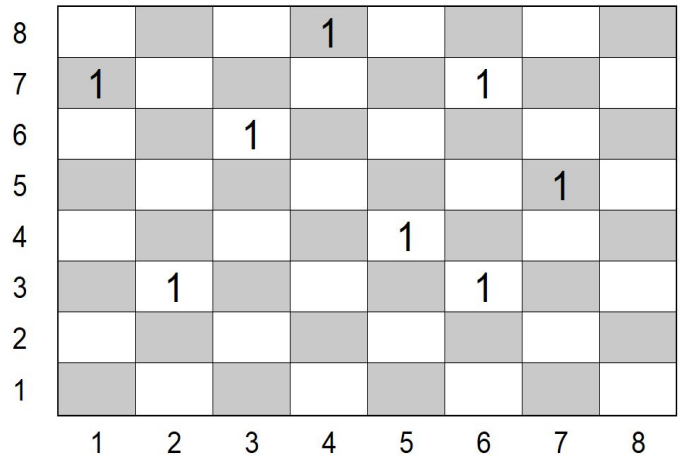


Figure 9: A Chess board .

The constraints which are related to the queen's move are given next.

1. Since in each row only one queen is allowed, we define the set in which each row contains only 1 one and the rest is 0. We call it the C_1 constraint .

$$C_1 = \{x \in \{0,1\}^{8\times 8} \mid \text{every row of } x \text{ is a unit vector}\} \tag{3.3}$$

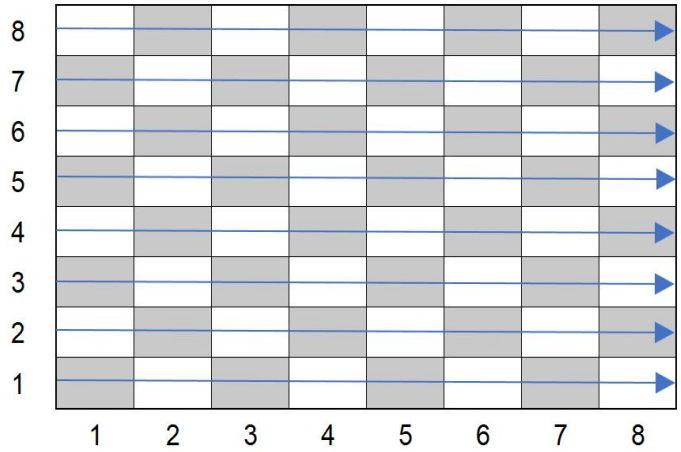


Figure 10: Row constraints.

2. Since in each column only one queen is allowed, we define the set in which each column contains only 1 one and the rest is 0. We call it the C_2 constraint

$$C_2 = \{x \in \{0,1\}^{8\times 8} \mid \text{every column of } x \text{ is a unit vector}\}. \tag{3.4}$$

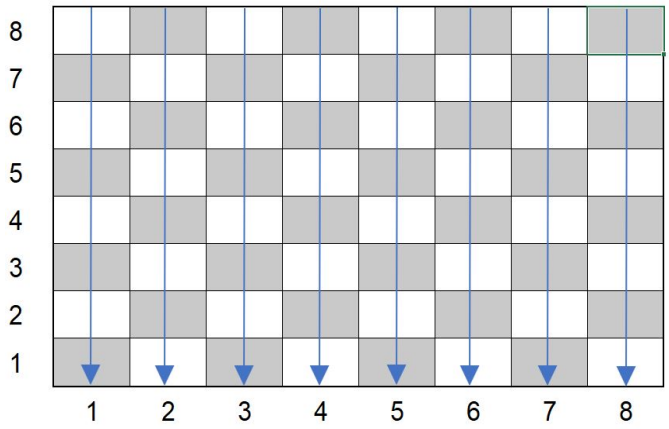


Figure 11: Column constraints.

3. Since in each diagonal only one queen is allowed, we define the set in which each diagonal contains only 1 one and the rest is 0. We call it the C_3 constraint. Observe that in a chess board there are 15 positive sloped diagonal, and only 8 queens, which mean that 7 of the 15 positive sloped diagonals are a zeros vectors.

$$C_3 = \{x \in \{0,1\}^{8\times 8} \mid \text{every positive diagonal of } x \text{ is a unit or zero vector}\} \tag{3.5}$$

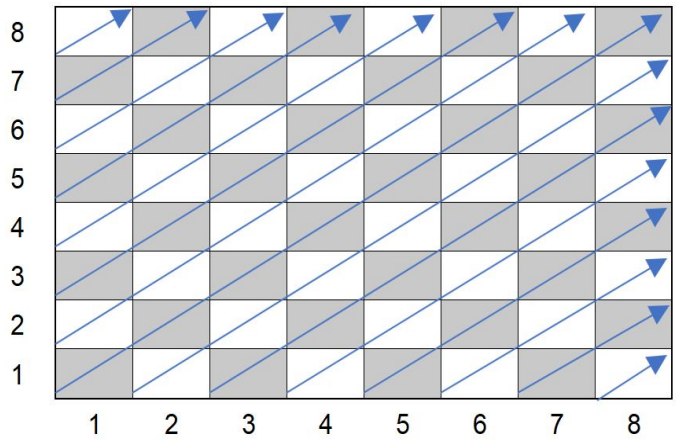


Figure 12: Positive slope diagonal.

4. The negative slope constraints are similar to the positive slope constraints, but in the negative direction. So define the constraint set:

$$C_4 = \{x \in \{0,1\}^{8\times 8} \mid \text{every negative diagonal of } x \text{ is a unit or zero vector}\} \tag{3.6}$$

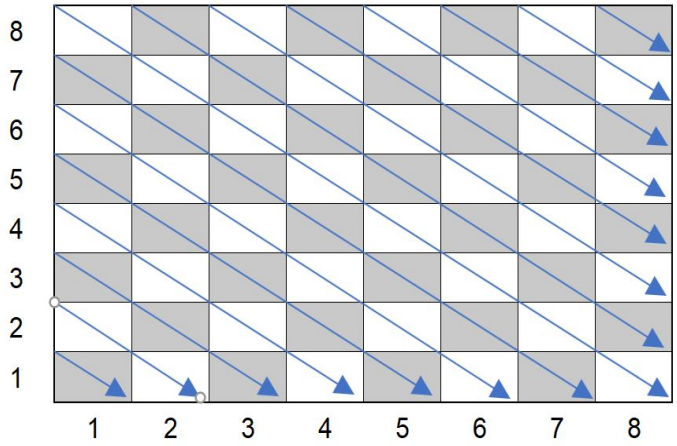


Figure 13: Positive slope diagonal.

In order to apply the Douglas–Rachford algorithm we first transform the problem into a continues form, that is $\{0,1\}^{8\times 8} \Rightarrow \mathbb{R}^{8\times 8}$ and show how the projection (and then reflections) onto the non-convex sets $C_1, \ldots, C_4$ can be easily implemented. This is mainly executed using Example 2.5. For example see projection of a point $x \in \mathbb{R}^{8\times 8}$ onto C_1.

$$\begin{pmatrix} 0.51 & 0.54 & 0.87 & 0.28 & 0.98 & 0.09 & 0.14 & 0.91 \\ 0.23 & 0.65 & 0.39 & 0.74 & 0.26 & 0.68 & 0.21 & 0.12 \\ 0.13 & 0.73 & 0.88 & 0.29 & 0.28 & 0.65 & 0.70 & 0.42 \\ 0.68 & 0.12 & 0.04 & 0.89 & 0.14 & 0.01 & 0.21 & 0.14 \\ 0.76 & 0.82 & 0.49 & 0.88 & 0.97 & 0.66 & 0.07 & 0.26 \\ 0.72 & 0.98 & 0.48 & 0.70 & 0.11 & 0.82 & 0.61 & 0.57 \\ 0.40 & 0.99 & 0.28 & 0.11 & 0.48 & 0.84 & 0.53 & 0.44 \\ 0.01 & 0.69 & 0.77 & 0.50 & 0.98 & 0.70 & 0.77 & 0.86 \end{pmatrix} \mapsto \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Figure 14: Projection onto C1.

Next we present some numerical examples. We generate randomly 8×8 matrix with 8 ones and then run the Douglas–Rachford for solving it. The results are presented in Figures 15–18 below.

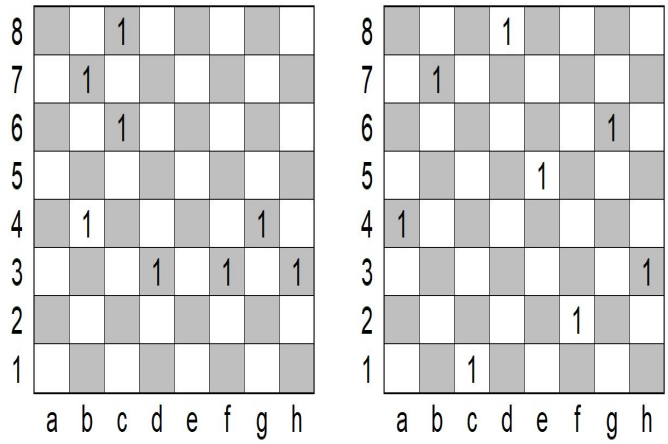


Figure 15: Solved in 154 iterations.

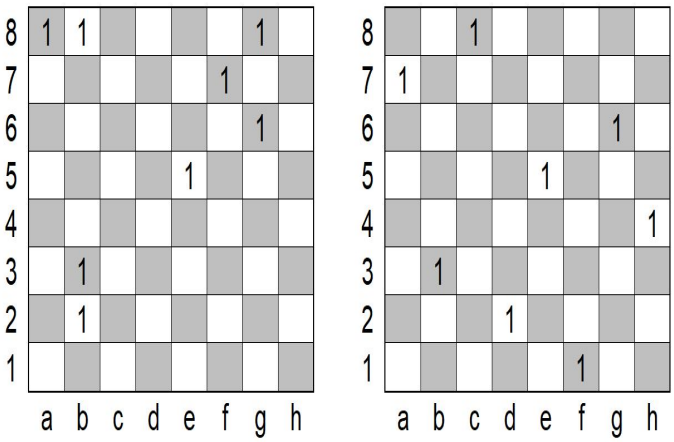


Figure 16: Solved in 117 iterations.

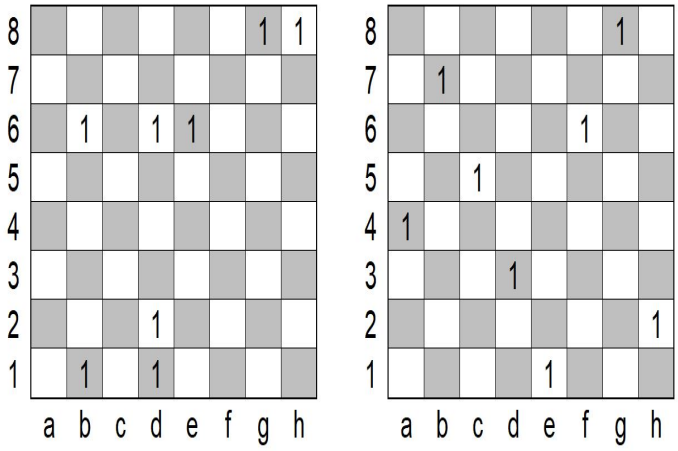


Figure 17: Solved in 188 iterations.

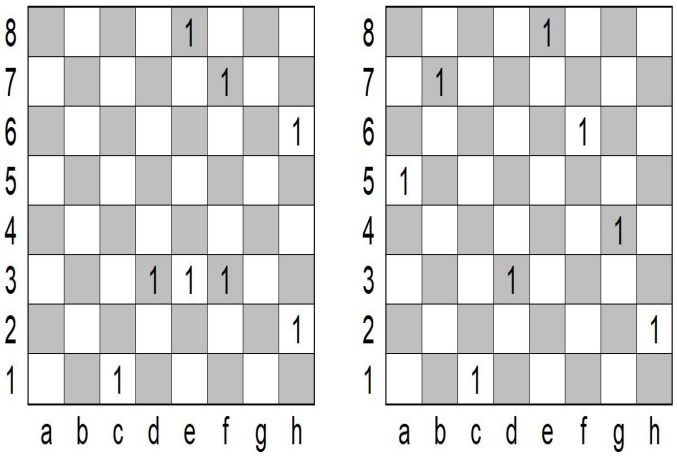


Figure 18: Solved in 48 iterations.

3.2.2 Sudoku

As a final non-convex implementation we consider the puzzle which is Sudoku. The Sudoku puzzle consists of a 9×9 grid and 3×3 subgrid that composes the grid. The objective is to fill a 9×9 grid with digits so that each column, each row, and each one of the nine 3×3 subgrids contain all of the digits from 1 to 9.

In order to remodel the problem we convert the 9×9 Sudoku matrix (call it A(i,j)) to a $9 \times 9 \times 9$ cube (call it Q(i,j,k)), by the rules:

$$Q(i,j,k) = \begin{cases} 1, & \text{if } k = A(i,j) \\ 0, & \text{otherwise.} \end{cases} \tag{3.7}$$

							1	
4								
	2							
				5		4		7
		8				3		
		1		9				
3			4			2		
	5		1					
			8		6			

Figure 19: A 9x9 Sudoku.

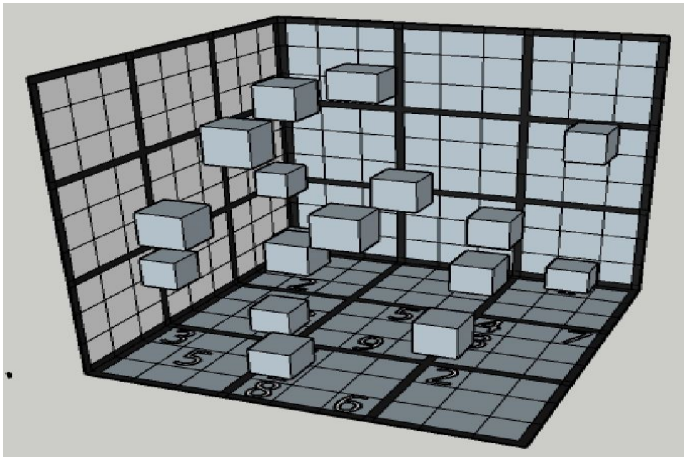


Figure 20: Representation of the above as a cubic Sudoku.

The constraints which are related to the Sudoku puzzle are given next.

1. This constraint ensure that there is no repetition numbers in the Sudoku's rows.

$$C_1 = \{Q \in \{0, \ldots, 9\}^{9^3} \mid Q(i, :, k) \text{ is a standard unit vector } \forall i, k\} \tag{3.8}$$

where $Q(i, :, k)$ means that the j coordinate is free.

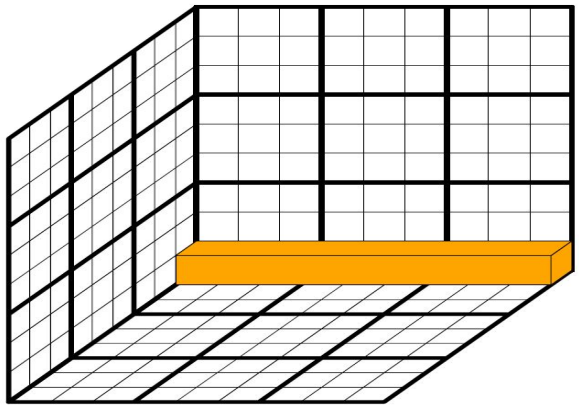


Figure 21: Row constraints.

2. This constraint ensure that there is no repetition numbers in the Sudoku's columns.

$$C_2 = \{Q \in \{0, \ldots, 9\}^{9^3} \mid Q(:, j, k) \text{ is a standard unit vector } \forall j, k\} \tag{3.9}$$

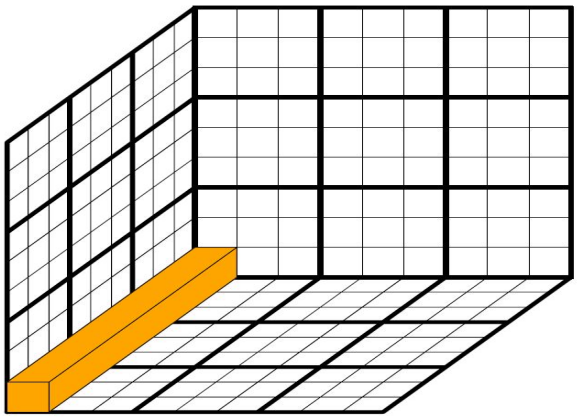


Figure 22: Column constraints.

3. This constraint ensure that there is no repetition numbers in the nine 3×3 subgrids.

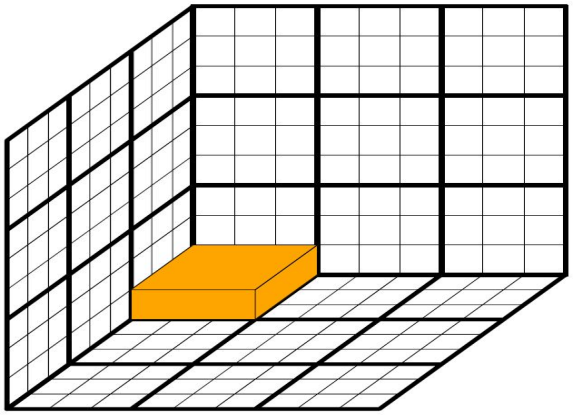


Figure 23: 3×3 constraints.

4. This constraint ensure that there is only 1 in every height.

$$C_3 = \{Q \in \{0, \ldots, 9\}^{9^3} \mid Q(i, j, :) \text{ is a standard unit vector } \forall j, k\} \tag{3.10}$$

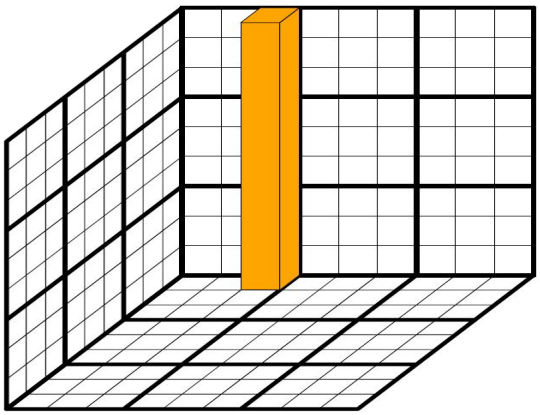


Figure 24: Height constraints.

5. This constraint ensure that the solution is compatible with the initial Sudoku.

$$C_4 = \{Q \in \{0,\ldots,9\}^{9^3} \mid Q(i,j,A(i,j)) = 1 \forall j,k\} \tag{3.11}$$

As before for the implementation of the Douglas–Rachford algorithm we transform the problem into a continues form, that is $\{0,\ldots,9\}^{9^3} \Rightarrow \mathbb{R}^{9^3}$ and show how the projection (and then reflections) onto the non-convex sets $C_1,\ldots,C_4$ can be easily implemented. As in the 8 queen puzzle, the projections are based on Example 2.5.

Next we present numerical illustrations for Sudoku collection taken from the internet.

							1	
4								
	2							
				5		4		7
		8				3		
		1		9				
3			4			2		
	5		1					
			8		6			

6	9	3	7	8	4	5	1	2
4	8	7	5	1	2	9	3	6
1	2	5	9	6	3	8	7	4
9	3	2	6	5	1	4	8	7
5	6	8	2	4	7	3	9	1
7	4	1	3	9	8	6	2	5
3	1	9	4	7	5	2	6	8
8	5	6	1	2	9	7	4	3
2	7	4	8	3	6	1	5	9

Figure 25: Solved in 974 iterations.

						1	2	
			3	5				
			6				7	
7						3		
			4			8		
1								
			1	2				
	8						4	
	5					6		

6	7	3	8	9	4	5	1	2
9	1	2	7	3	5	4	8	6
8	4	5	6	1	2	9	7	3
7	9	8	2	6	1	3	5	4
5	2	6	4	7	3	8	9	1
1	3	4	5	8	9	2	6	7
4	6	9	1	2	8	7	3	5
2	8	7	3	5	6	1	4	9
3	5	1	9	4	7	6	2	8

Figure 26: Solved in 1915 iterations.

							1	2
		3	6					
					7			
4	1			2				
			5			3		
7						6		
2	8						4	
			3			5		

6	7	9	8	3	5	4	1	2
1	2	3	6	9	4	7	5	8
5	4	8	2	1	7	9	3	6
4	1	6	7	2	3	8	9	5
8	9	2	5	6	1	3	7	4
7	3	5	4	8	9	6	2	1
2	8	7	9	5	6	1	4	3
9	6	1	3	4	2	5	8	7
3	5	4	1	7	8	2	6	9

Figure 27: Solved in 686 iterations.

						1	2	
		8		3				
						4		
1	2		5					
					4	7		
	6							
5		7				3		
			6	2				
			1					

3	4	6	7	9	5	8	1	2
2	5	8	4	3	1	6	9	7
9	7	1	8	6	2	5	4	3
1	2	9	5	7	6	4	3	8
8	3	5	2	1	4	7	6	9
7	6	4	3	8	9	2	5	1
5	1	7	9	4	8	3	2	6
4	9	3	6	2	7	1	8	5
6	8	2	1	5	3	9	7	4

Figure 28: Solved in 374 iterations.

							1	2
	4			5				
					9			
	7		6			4		
			1					
							5	
				8	7	5		
6		1				3		
2								

5	9	8	4	6	3	7	1	2
7	4	2	8	5	1	6	3	9
3	1	6	7	2	9	8	4	5
1	7	5	6	3	2	4	9	8
8	6	9	1	4	5	2	7	3
4	2	3	9	7	8	1	5	6
9	3	4	2	8	7	5	6	1
6	8	1	5	9	4	3	2	7
2	5	7	3	1	6	9	8	4

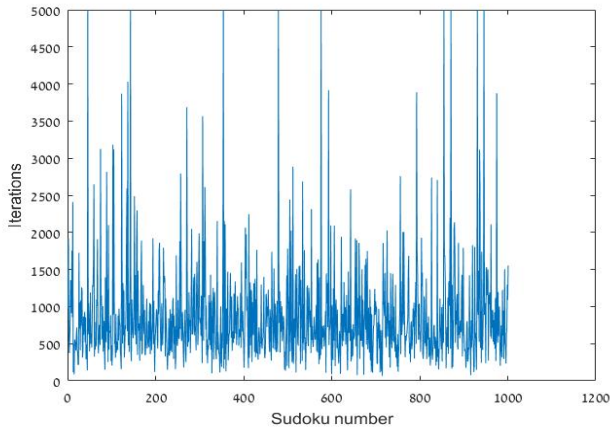
Figure 29: Solved in 556 iterations.

							1	2
	5		4					
							3	
7			6			4		
		1						
				8				
9	2					8		
			5	1		7		
					3			

3	6	4	9	7	8	5	1	2
1	5	2	4	3	6	9	7	8
8	7	9	1	2	5	6	3	4
7	3	8	6	5	1	4	2	9
6	9	1	2	4	7	3	8	5
2	4	5	3	8	9	1	6	7
9	2	3	7	6	4	8	5	1
4	8	6	5	1	2	7	9	3
5	1	7	8	9	3	2	4	6

Figure 30: Solved in 829 iterations.

Finally we show the number of iteration needed for solving a collection of 1000 Sudoku puzzles. The average number of iterations is 879 iterations.



4 Conclusions

In this project we are concern with feasibility problems and problems that can be remodeled in such a way, convex and non-convex. We implement the Douglas-Rachford algorithm in an appropriate product space for solving these problems and shows its efficiency.

This project suggests that in some cases remodeling the problem can yield better results than adjusting the algorithms. such an approach is the motivation of the newly introduced Superiorization methodology (http://math.haifa.ac.il/YAIR/bib-superiorization-censor.html) which is and can be used for further research.

References

[1] F. J. Aragón Artacho, J. M. Borwein and M. K. Tam, Recent results on Douglas–Rachford methods for combinatorial optimization problems, *Journal of Optimizaion Theory and Applications* **163** (2014), 1–30.

[2] F. J. Aragón Artacho, J. M. Borwein and M. K. Tam, Global behavior of the Douglas–Rachford method for a nonconvex feasibility problem, *Journal of Global Optimization* **65** (2016), 309–327.

[3] F. J. Aragón Artacho, Y. Censor and A. Gibali, The Cyclic Douglas–Rachford algorithm with $\boldsymbol{r}$-sets-Douglas–Rachford Operators, accepted for publication in *Optimization Methods and Software*, 2018.

[4] H. H. Bauschke and J. M. Borwein, On projection algorithms for solving convex feasibility problems, *SIAM Review* **38** (1996), 367–426.

[5] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, Second Edition, Springer International Publishing AG, 2017.

[6] J. M. Borwein and M. K. Tam, A cyclic Douglas–Rachford iteration scheme, *Journal of Optimizaion Theory and Applications* **160** (2014), 1–29.

[7] J. Douglas and H. H. Rachford, On the numerical solution of the heat conduction problem in two and three space variables, *Transactions of the American Mathematical Society* **82** (1956), 421–439.

[8] D. R. C. García, *Contributions to the Theory and Applications of Projection Algorithms*, PhD. thesis, December 2018.

[9] P. L. Lions and B. Mercier, Splitting algorithms for the sum of two nonlinear operators, *SIAM J. Numer. Anal.* **16**(1979), 964–979.