Annals of University of Craiova, Math. Comp. Sci. Ser.
Volume 36(1), 2009, Pages 63–68
ISSN: 1223-6934

Partial simulation of (CQ) algorithm

Cristina Popîrlan

Abstract. This paper presents a partial simulation of (CQ) algorithm of Nakajo and Takahashi. The algorithm implementation permits us to analyze the sets C, Q and $C \bigcap Q$. This way we can see the importance of the weight factor in determination of the next iteration. The application represents one step of the (CQ) algorithm, but if we can calculate the projection of the initial point on the set $C \bigcap Q$, then we can generate the entire algorithm.

2000 Mathematics Subject Classification. Primary 65F10; Secondary 65Y10.
Key words and phrases. (CQ) algorithm, iterative methods, Mann iteration.

1. Introduction

The (CQ) algorithm was introduced in 2003 by Nakajo and Takahashi and it is presented in the following formulas.

$$\begin{cases} x_0 = x \in Q, \\ y_n = (1-t_n)x_n + t_n T x_n, \\ C_n = \{z \in C : ||y_n - z|| \le ||x_n - z||\}, \\ Q_n = \{z \in C : \langle x_0 - x_n, x_n - z\rangle \ge 0\}, \\ x_{n+1} = P_{C_n \bigcap Q_n} x_0, \end{cases}$$

where $P_M x$ is the projection of x on the set M.

The main results of [1] is:

Theorem 1.1. [1]

Let $C \subset \mathcal{H}$ be a closed convex set from a Hilbert space $\mathcal{H}$ and $T : C \to C$ a nonexpansiv operator with $Fix(T) \neq \emptyset$. Suppose that the weight sentence $\{t_n\}_{n\ge 0}$ satisfies the condition $t_n \in (0,1]$.

Then the sequence $\{x_n\}_{n\ge 0}$ generated by the (CQ) algorithm converges strongly to $P_{Fix(T)}x_0$.

The (CQ) algorithm is also strong convergent if the operator is k-demicontractiv [2] and the following condition is satisfied: the operator I-T is demiclosed at zero.

The algorithm was intensely studied in order to analyze his convergence and for this reason a lot of generalizations of the (CQ) algorithm were given. For example, in a recent paper, Takahashi, Takevchi and Kubota [3] introduced a new hybrid method for a family of nonexpansive operators. When the family is reducing to a single operator, with a given initial iteration, $x_0 \in C$, and taken $C_1 = C$ and $u_1 = P_{C_1}x_0$, then the algorithm is defined by the following formulas:

$$\begin{cases} y_n = t_n u_n + (1-t_n)Tu_n, \\ C_{n+1} = \{z \in C_n : ||y_n - z|| \le ||u_n - z||\}, \\ u_{n+1} = P_{C_{n+1}} x_0, \end{cases}$$

where the weight sentence $\{t_n\}_{n\ge 0}$ satisfies the condition $0 \le t_n \le t < 1$.

Remark 1.1. *The concrete application of the (CQ) algorithm implies the evaluation of the sets C_n and Q_n and then the explicit calculation of the projection of x_0 on the intersection $C_n \bigcap Q_n$, or to solve a conditioned optimization problem, at ever step of the iteration. The papers which contains data about the (CQ) algorithm implementation are limited.*

It can be shown that the sets C_n and Q_n are closed and convex, so the set $C_n \bigcap Q_n$ is also closed and convex, and we can say that the sequence x_{n+1} is well defined.

We also observe that $Fix(T) \subset C_n \bigcap Q_n$ and so the construction of the sets C_n and Q_n is natural; if $z \in Fix(T)$, then, if T is demicontractiv, then the relation that defines C_n is satisfied, and if $z \in Fix(T)$ and x_n is the projection of x_0 on Q_n, then the relation that defines Q_n is also satisfied. So it is natural to take x_{n+1} as close to the intersection and so we take

$$x_{n+1} = P_{C_n \bigcap Q_n} x_0.$$

2. The (CQ) algorithm - application

The main idea of the algorithm is to project each Mann iteration on an intersection of sets that are built at every step C_n and Q_n.

The application simulate a step from the algorithm and represents the sets C_n and Q_n. For different input parameters, the weight factor t and the network that is analyzed, we study how the network is organized between the sets C, Q and $C \bigcap Q$.

The program analyses the repartition of the points from a square which has the $2a$ dimensions, the square has the center in axes origin, where $a > 2$ is given by the user of the application. In the square is built a n^2 points network, where n is also given by the user. For every point from the network it is study if the points belongs to one, two or none of the sets C and Q. This sets are different colored for each case in part:

(1) **blue** for the points in C that are not in Q;
(2) **green** for the points that are not in C but are in Q;
(3) **red** for the points in C and Q;
(4) **yellow** for the points that are not in C or in Q.

The application starts by establishing the dimension of the square where the network is defined. After that, the user gives the dimension of the network, more exactly he decide how big the network is, n^2 points in the network.

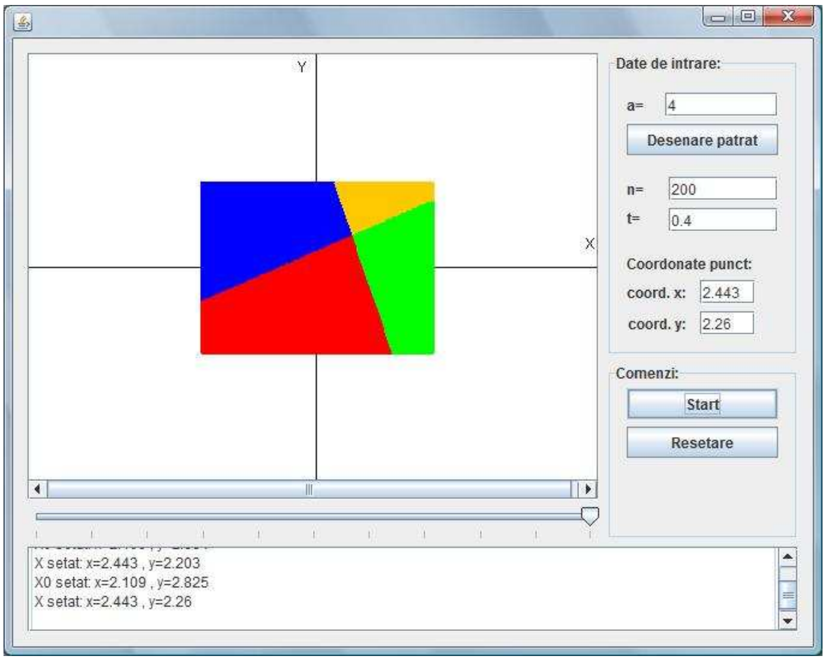
Next is given some very important parameters an that are the weight factor and the initial points (the starting points and the current iteration). The application allows the user to change the weight factor and to analyze the distribution of the points from the network for the same initial values.

We consider the following four inequations (that simulate a square of dimension 1 with the center in axes origin) on which we will make the projections from the algorithm:

$$\begin{cases} d_1(x_1,x_2) = x_1 + 1 >= 0 \\ d_2(x_1,x_2) = -x_1 + 1 >= 0 \\ d_3(x_1,x_2) = x_2 + 1 >= 0 \\ d_4(x_1,x_2) = -x_2 + 1 >= 0 \end{cases}$$

If we consider $x = (x_1, x_2) \in \mathcal{R}^2$, we check for each one of the inequations if $d_i(x) < 0$, $i = \overline{1,4}$, and we take k so that

$$d_k(x) = \max_{i=\overline{1,4}, d_i(x)<0} |d_i(x)|.$$



Date de intrare:

a= 4

n= 200

t= 0.4

Coordonate punct:

coord. x: 2.443

coord. y: 2.26

Comenzi:

```
X setat: x=2.443 , y=2.203
X0 setat: x=2.109 , y=2.825
X setat: x=2.443 , y=2.26
```

FIGURE 1. Simulation of (CQ) algorithm

We consider the operator T from the (CQ) algorithm to be the projection of x on d_k.

```
int k=calcDrProiectie();
Dreapta D=DH1.getDreapta(k);
Punct P=DH1.getX1();
double x=P.x-D.geta()*((D.geta()*P.x+D.getb()*P.y+D.getc())/
  (Math.pow(D.geta(), 2)+Math.pow(D.getb(),2)))*DH1.gett();
double y=P.y-D.getb()*((D.geta()*P.x+D.getb()*P.y+D.getc())/
  (Math.pow(D.geta(), 2)+Math.pow(D.getb(),2)))*DH1.gett();
DH1.setY(x, y);
```

The application is built around the following idea: there are given two points one x_0 that represents the initial point and one that represents the current iteration. It is obvious that we do not exactly execute the (CQ) iteration, but we make a step from the iteration with a point given by the user. If at the next step we take, by approximation, the projection of the current iteration on to the set $C \bigcap Q$ then we can simulate the (CQ) algorithm. So if we consider the possibility of recalculating and reiterating but using the same x_0 and taking another x then we can simulate the (CQ) algorithm. The application can be improved by considering the next x in the position of the projection of x_0 on the sets intersection.

x_0 and x are given by the user, and y is calculated by the following formula $y = (1-t)x + tT(x)$, where T is the projection constructed before. z is the point that browses the entire network. The application browses the points network and represents those three sets. In consequence, we can see the visual representation of the sets C, Q and $C \bigcap Q$.

```
try {
```

```
int n=DH1.getn();
int k=-1;
for (int i=0;i<=n;i++)
  for (int j=0;j<=n;j++)
  {
    k++;
    TestZ(k);
    DH1.setDrawPoz(k);
    jPD.repaint();
    jTP.setText(jTP.getText()+"Z"+k+":
       z1="+DH1.getPunctZ(k).x+" ,
       z2="+DH1.getPunctZ(k).y+"\n");
    Thread.sleep(sleepValue);
   }
  } catch (InterruptedException ex) {
    Logger.getLogger(Executor.class.getName()).
          log(Level.SEVERE, null, ex);
       }
```

Running the application we observe four categories of points:

(1) points that belong only to C;
(2) points that belong only to Q;
(3) points that belong to C and Q;
(4) points that does not belong to C or to Q.

Next we analyze the organization of this points in sets considering different initial point and current iteration:

(1) if the two given points (x_0 and the current iteration) are in the same dial, but they are not in the square, then the points from the network are colored only in red and blue no matter what the value of the weight factor is;
(2) if the two given points (x_0 and the current iteration) are in the same dial, but they are in the square, then the points from the network are colored in red, blue and no matter what the value of the weight factor is;
(3) if the two given points (x_0 and the current iteration) are in the same dial, but the firs is inside the square and the second is outside the square then the points from the network are colored in yellow and blue no matter what the value of the weight factor is, with the observation that for certain values of the points it appears and a number of points that turn red but their number is relatively small compared to the total number points in the network;
(4) if the two given points (x_0 and the current iteration) are in the same dial, but the firs is outside the square and the second is inside the square then the points from the network are colored in red, green and blue no matter what the value of the weight factor is. The exception is represented by the points from the third dial (the points $x = (x_1, x_2)$, with $x_1 < 0$ and $x_2 < 0$) for which the network is colored only in red and blue, with the observation that for certain values of the points it appears and a number of points that turn green but their number is relatively small compared to the total number points in the network;
(5) for all the other cases the network is colored in all four colors. The distribution of those four colors depends on the weight factor.

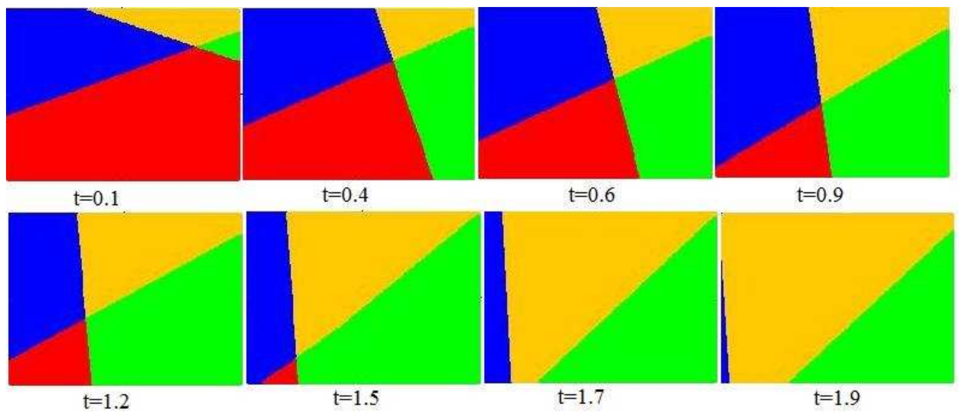


FIGURE 2. The influence of the weight factor over the distribution network

We can thus draw the conclusion that control factor largely influences the distribution of network points in the two sets C and Q, and also the entries of user position puts her mark on how these points in the network are colored.

Below is how they influence factor control of network distribution points.

Example 2.1. *We consider the following situation:*

- $a = 4$;
- *n big enough;*
- $x_0 = (1.608, 0.565)$;
- $x = (-0.355, -1.469)$.

The analysis of this example is made in Figure 2.

Example 2.2. *We consider the following situation:*

- $a = 4$;
- *n big enough;*
- $x_0 = (0.397, -1.356)$;
- $x = (-1.482, 0.339)$.

The analysis of this example is made in Figure 3.

Remark 2.1. *Note that for $t = 0.1$ the number of color points red is much larger than the color points green, and as the T increases this difference begins to decrease, so that for $t = 2$ is the situation reverse, the number of points colored in red is much smaller than the points colored in green.*

From the examples presented it can be concluded that the weight factor largely influences distribution points in the network.

We noticed that there are some cases where blue predominates points (this means that the points of the network is in the crowd C and not in crowds Q), if the user selected points are like $(negative, positive)$.

An interesting observation was that whatever the factor of control and whatever the initial values of user data have colored dots colored in blue and red dots, so in any situation there are only points in C and there are points is in both sets C and Q.

3. Conclusions and future work

This work included the following aspects:

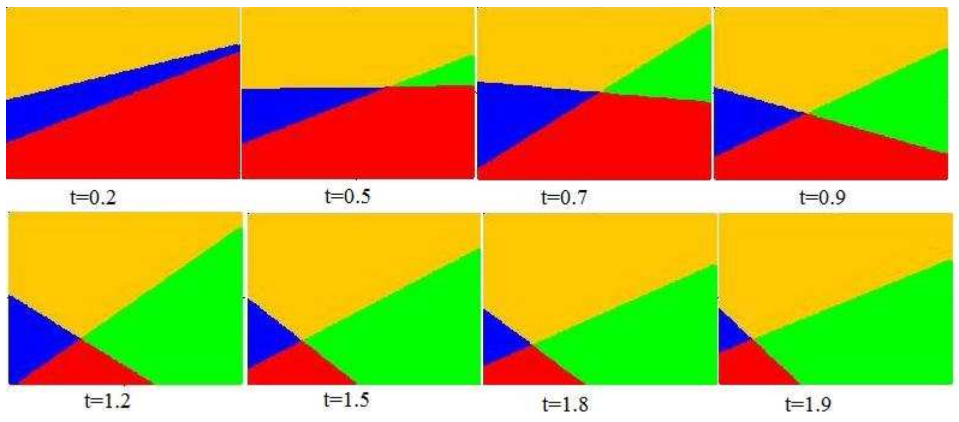


FIGURE 3. The influence of t over the distribution network

- presentation of the mathematical (CQ) algorithm;
- the description of the application that simulates the (CQ) algorithm;
- an analysis of the influence of the weight factor on the network distribution.

As a future work we propose to extend the simulation in order to do a real implementation of the (CQ) algorithm.

References

[1] K. Nakajo, W. Takahashi :Strong convergence theorems for nonexpansive mappings and nonexpansive semigroups, Journal of Mathematical Analysis and Applications, 279:372379, 2003
[2] G. Marino, H.K. Xu :Weak and strong convergence theorems for strictly pseudocontraction in Hilbert spaces, Journal of Mathematical Analysis and Applications, 329:336349, 2007
[3] W. Takahashi, Y. Takeuchi, R. Kubota: Strong convergence theorems by hybrid methods for families of nonexpansive mappings in Hilbert spaces, Journal of Mathematical Analysis and Applications, 341:276286, 2008

(Cristina Popîrlan) UNIVERSITY OF CRAIOVA, ALEXANDRU IOAN CUZA 13, 200585 CRAIOVA, ROMANIA
E-mail address: `cristina_popirlan@yahoo.com`