Evaluation of algorithms to measure a psychophysical threshold using digital applications

Silvia Bonfanti[1 [0000−0001−9679−4551]] and Angelo Gargantini[1 [0000−0002−4035−0131]]

Department of Management, Information and Production Engineering, University of Bergamo, Bergamo, Italy
{silvia.bonfanti,angelo.gargantini}@unibg.it

Abstract. The use of digital applications like in mobile phone or on the web to perform psychophysical measurements led to the introduction of algorithms to guide the users in test execution. In this paper we show four algorithms, two already well known: STRICTN and PEST, and a two that we propose: PESTN and BESTN. All the algorithms aim at estimating the level of a psychophysical capability by performing a sequence of simple tests; starting from initial level N, the test is executed until the target level is reached. They differ in the choice of the next steps in the sequences and the stopping condition. We have simulated the application of the algorithms and we have compared them by answering a set of research questions. Finally, we provide guidelines to choose the best algorithm based on the test goal. We found that while STRICTN provides optimal results, it requires the largest number of steps, and this may hinder its use; PESTN can overcome these limits without compromising the final results.

1 Introduction

The use of computers and digital applications to perform psychophysical measurements has given rise to several automatic procedures to be applied. The objective of these procedures is to determine as rapidly and precisely as possible the value of a psychophysical variable.

In the paper, we focus on estimating psychophysical thresholds by providing a sequence of simple tasks to the patients. Following the classification of methodological for psychophysical evaluation proposed in [12], we can consider three parameters: the task of an observer to judge, stimulus arrangement, and statistical measure. Based on the proposed classification, the set of algorithms taken into account for the comparison fall into the following groups. Regarding the task, we assume that the observer's task is the *classification* of some type. The observer, once a stimulus has been presented, has to judge if some attribute or aspect is present or absent or to classify the stimulus. Regarding the stimuli to be presented, we assume that are *fixed*, i.e. they do not vary during the time they are being observed[1]. Usually, of course, they are varied between observations. Regarding the measure of stimulus, we assume that a level is associated with every observation and this level is used to estimate the psychophysical threshold.

[1] This assumption could be relaxed provided that the classification of the stimulus is meaningful

Furthermore, in our case study, we assume that the psychophysical of the user could be null as a result of missing capability by the user and the test should discover that.

Before the introduction of digital technologies, psychophysical measurements were made by using simple devices or printed paper cards and the observer had to judge the responses. The observer guided the test procedure that could be partially fixed based on test execution. Nowadays, tests are becoming more computerized, partially automatized and the observer may have only partial control during the test execution. The test execution and the estimation of the psychophysical threshold are decided by an algorithm that should correctly diagnose the level of the measured parameter, by minimizing the number of false positive/negative.

The advantages are that the observer cannot interfere with the testing process and the results can be objectively validated. However, there is the risk that the algorithms are not precise or they are not as efficient as the observer would be thanks to the experience in providing these tests. For this reason, in this paper, we present and compare three algorithms that could be used for psychophysical measurements. We assume that the desired psychophysical capability can range from a max value to a minimal (lower) level. The test starts at level N and finishes when the user reaches his best capacity level [2].

The presented work is an extension of the paper [2]. We have extended the paper by introducing a new algorithm (the BESTN algorithm) and we have compared its performance with those of the already presented algorithms. Moreover, we have implemented a new scenario for the simulation in which we have considered that during a test (especially if it is long) the *attention* of the patient decreases.

The paper is organized as follows. In Sect. 2 we present the algorithms and the simulation protocol is introduced in Sect. 3. In Sect. 4 we answer a set of research questions about the features of the algorithms and we provide guidelines to choose the algorithm based on the test goal in Sect. 5. Related works are reported in Sect. 6.

2 Algorithms

In this section, we present the algorithms we have implemented to measure a psychophysical threshold: STRICTN, PEST, both well known in the literature and widely used, and two new ones, PESTN and BESTN, that try to improve the performances of the previous two.

All the proposed algorithms start from the assumptions that there are several *levels* of a given psychophysical capability, every person has a different *threshold* of such psychophysical capability, and the algorithm must evaluate this threshold. The basic idea behind all the proposed algorithms is the following. The test starts at *init* level, which corresponds to the easiest level (decided by the observer), and it is changed until the person is no longer able to answer correctly. The best reachable level that the person can achieve is called *target* and it corresponds to the most difficult level of the test. In this paper we assume that the *init* is greater than *target*, so for instance, the test starts from 10 and must reach 1 to measure the psychophysical capability. The level

[2] The data and materials for all experiments are available at `https://github.com/silviabonfanti/3d4ambAlgorithms.git`

is therefore decremented until it reaches the minimum possible. However, not all the subjects can reach the *target* since some may have a limited psychophysical capability. The problem the algorithms try to address is to find the *threshold* of the single person, that we assume to between *init* and the *target*. To find such threshold, at every level the person is asked to guess the right answer at that given level, and after that the algorithm proceeds depending on the correctness of the answer. When the user finishes the test, the result can be: 1. PASSED at level X: the user has passed successfully the test and his psychophysical capability *threshold* is certified at level X. 2. FAILED: the user did not pass the test because the algorithm has found that he does not have the psychophysical capability.

The algorithms differ from one another in the following aspects: 1. when the user guesses right but the *target* i snot reached, what is the next tested level; 2. the number of right answers given at the *target* level to be certified; 3. the error management when the user does not guess the right answer; 4. the policy to interrupt the test and certify or not the level.

All the algorithms are explained in the next sections. All can be generalized in case the tests are performed using a different scale of levels, for instance by starting to 1 and going to a maximum value. Moreover, it is possible to start the test from a level which is not the easiest one.

2.1 STRICTN

The strict staircase algorithm, STRICTN, shown in Fig. 1, is well known in the literature and widely used (for instance [18]) since it can measure precisely the psychophysical threshold. However its main disadvantage is that it takes a lot of time, mostly when the difference between the starting threshold and target threshold is high. The test starts at the starting threshold `initThreshold`. If the user guesses the answer (`answer=RIGHT`) the `currentThreshold` (the threshold currently under test) is decremented. The algorithm stops in *PASSED* state when the `targetThreshold` is reached (`currentThreshold = targetThreshold`) and the user answers correctly N times, where N is equals to 3 or to a value chosen at the beginning of the test (`rightAnswersToCertify`). If the answer is `WRONG` the threshold is re-tested, if another error is performed the threshold is incremented and it becomes the new target (only higher levels can be certified at this point). A threshold is *PASSED* if the user responds correctly N times at that threshold. In the event that the person is not able to answer correctly N times the test result is *FAILED*.

The STRICTN algorithm generally requires long trials. To overcome the disadvantage of this, we have introduced the BESTN algorithm, explained in the next section, which reduces the number of right answers requested to certify a level.

2.2 BESTN

The key concept behind BESTN algorithm (see Algorithm 2) is that the user has to answer `rightAnswerToCertify` times correctly in the same threshold or in two consecutive thresholds. Moreover, this algorithm set the maximum answers (`RIGHT` or `WRONG`) possible for each threshold which is set to `answersToCertify` times. The test starts

Algorithm 1: STRICTN

```
Input  : initThreshold, targetThreshold, rightAnswersToCertify
Output: currentResult, currentThreshold
currentThreshold = initThreshold;
do
    /* store the number of RIGHT/WRONG answers for each threshold                    */
    getAndStoreAnswer(currentThreshold, answer);
    switch answer do
        case RIGHT do
            if currentThreshold>targetThreshold then
                currentThreshold - -;
                currentResult = CONTINUE;
            else
                /* compare the number of RIGHT answers at current threshold         */
                if getNumAnswers(currentThreshold, RIGHT) >= rightAnswersToCertify then
                    currentResult = PASSED;
                else
                    currentResult = CONTINUE;
                end
            end
        case WRONG do
            /* compare the number of WRONG answers at current threshold             */
            if getNumAnswers(currentThreshold, WRONG) >= WRONG_TO_STOP then
                if currentThreshold < initThreshold then
                    currentThreshold ++;
                    targetThreshold = currentThreshold;
                    currentResult = CONTINUE;
                else
                    currentResult = FAILED;
                end
            else
                currentResult = CONTINUE;
            end
    end
while currentResult == CONTINUE;
return [currentResult, currentThreshold]
```

Algorithm 2: BESTN

```
Input  : initThreshold, targetThreshold, rightAnswersToCertify, answersToCertify
Output: currentResult, currentThreshold
currentThreshold = initThreshold;
do
    getAndStoreAnswer(currentThreshold, answer);
    switch answer do
        case RIGHT do
            if currentThreshold > targetThreshold then
                currentThreshold - -;
                currentResult = CONTINUE;
            else if (getNumAnswers(currentThreshold, RIGHT) + getNumAnswers(currentThreshold+1,
              RIGHT) >= rightAnswersToCertify) then
                currentResult = PASSED;
            else
                currentResult = CONTINUE;
            end
        case WRONG do
            if getNumAnswers(currentThreshold, WRONG) == WRONG_TO_STOP then
                if currentThreshold < initThreshold then
                    currentResult = CONTINUE;
                    currentThreshold ++;
                    targetThreshold = currentThreshold;
                else
                    currentResult = FAILED;
                end
            else if getNumAnswers(currentThreshold) == answersToCertify then
                if currentThreshold < initThreshold then
                    if (getNumAnswers(currentThreshold, RIGHT) +
                      getNumAnswers(currentThreshold+1, RIGHT) >= rightAnswersToCertify) then
                        currentResult = PASSED;
                    else
                        currentResult = CONTINUE;
                        currentThreshold ++;
                        targetThreshold = currentThreshold;
                    end
                else
                    currentResult = FAILED;
                end
            else
                currentResult = CONTINUE;
            end
    end
while (currentResult = CONTINUE);
return [currentResult, currentThreshold]
```

at initThreshold, if the person answer is RIGHT and the currentThreshold does not correspond to the targetThreshold, the threshold is decremented and the test CONTINUEs. In case the current threshold corresponds to the threshold to be certified (currentThreshold=targetThreshold) there are two different options:

- the user has answered correctly rightAnswerToCertify times in the current threshold (if current threshold is the lower certifiable) or the user has answered correctly answersToCertify times in two consecutive thresholds (current threshold and previous threshold): the test finishes in PASSED state.
- the user has not answer correctly enough times, so the test CONTINUE.

In case the user has answered WRONG enough times (WRONG_TO_STOP) at the same threshold and the current threshold is lower than the maximum certifiable threshold (initThreshold), the user can CONTINUE the test, the targetThreshold is set equals to the currentThreshold. Nevertheless, if the current threshold is equal to the initThreshold, the test stops and the test FAILED. When the second answer to current threshold is wrong but the user has previously answered correctly rightAnswerToCertify the test finishes PASSED. If the second answer to current threshold is wrong and the previous conditions are not satisfied, the test CONTINUEs, the threshold is incremented and the new target threshold is equal to the currentThreshold. Finally, if the user answers answersToCertify times at current threshold (one correctly and one not) and the current threshold is equals to maxThreshold, the test finishes FAILED.

Although the required steps are decreased due to the improvement made compared to STRICTN algorithm, it still high. For this reason, the PEST algorithm, explained in the next section, has been introduced in the past, with the aim of reducing the number of required steps.

2.3 PEST

PEST (Parameter Estimation by Sequential Testing) algorithm (see Algorithm 3) has been proposed in [13]. This algorithm belongs to the adaptive methods family which are modified according to the moment-by-moment responses. The goal of PEST is to identify the psychophysical threshold with a minimum number of possible steps. The test starts at initThreshold and the goal is to reach the targetThreshold, the most difficult. Threshold of the test are into a window bounded by a left limit limitL and a right limit limitR. Initially, the variables limitL and limitR are set respectively to the starting level initThreshold and the targetThreshold. If the user answer is RIGHT the left limit is set to the current threshold and in the next step the tested threshold is equals to the round downward the mean between limitL and limitR to its nearest integer. The test continues until limitL and limitR correspond, the test finishes in PASSED state at current threshold. If the user answer is WRONG, the right limit is set to the current threshold and in the next step the tested threshold is equals to the round upward the mean between limitL and limitR to its nearest integer. Also in this case, the test continues until *limitL* and *limitR* correspond or currentThreshold reaches initThreshold, but if the user answers wrongly twice at currentThreshold the test finishes in *FAILED* state.

Algorithm 3: PEST

```
input  : initThreshold, targetThreshold
output: currentResult, currentThreshold
currentThreshold = initThreshold;
limitL = currentThreshold; limitR = targetThreshold;
chance = 2;
do
    getAndStoreAnswer(currentThreshold, answer);
    switch answer do
        case RIGHT do
            limitL = currentThreshold;
            currentThreshold = floor((limitL + limitR) / 2);
            if limitL == limitR then
                currentResult = PASSED; currentThreshold = limitL;
            else
                currentResult = CONTINUE;
            end
        case WRONG do
            if currentThreshold == initThreshold OR limitL == limitR then
                if chance > 0 then
                    chance–; currentResult = CONTINUE;
                else
                    currentResult = FAILED;
                end
            else
                limitR = currentThreshold;
                currentThreshold = ceil((limitL + limitR) / 2);
                currentResult = CONTINUE;
            end
    end
while (currentResult = CONTINUE);
return [currentResult, currentThreshold]
```

Compared to BESTN and STRICTN the number of steps required for PEST is significantly decreased, but at the end of the test, we are not sure that the certified threshold is the real threshold owned by the user. This is because the PEST algorithm requires only one correct answer to certify the target threshold, and it can be right just for randomness. For this reason, we have improved the PEST algorithm as explained in the next section.

Algorithm 4: PESTN

```
input  : initThreshold, maxThreshold, targetThreshold, rightAnswersToCertify
output: currentResult, currentThreshold
do
    if firstPhase then
        if answer == WRONG then
            if answers[limitL - 1] == 0 && currentThreshold == maxThreshold then
                answers[limitL - 1]–;
            else if answers[limitL - 1] == -1 && currentThreshold==maxThreshold then
                currentResult=FAILED;
            else
                limitR = currentThreshold; limitsOneStep();
                currentThreshold = ceil((limitL+limitR)/2);
                answers[limitR - 1] - -;
            end
        else if answer == RIGHT then
            limitL = currentThreshold; limitsOneStep();
            currentThreshold = floor((limitL+limitR)/2);
            answers[limitL - 1] ++;
        end
    else
        if answer == RIGHT then
            answers[currentThreshold - 1] ++;
        else if answer == WRONG then
            answers[currentThreshold - 1] -= weight; weight = weight * 3;
        end
        if answers[currentThreshold - 1] >= rightAnswersToCertify then
            currentResult = PASSED;
        else if (answers[currentThreshold - 1] <= -2) & (currentThreshold < maxThreshold) then
            weight = 1; currentThreshold ++;
        else if (answers[currentThreshold - 1] <= -2) & (currentThreshold == maxThreshold) then
            currentResult=FAILED;
        end
    end
while (currentResult = CONTINUE);
return [currentResult, currentThreshold]

Function limitsOneStep:
    if (limitL - limitR) == 1 then
        firstPhase = false; currentThreshold = limitR;
        if limitR != 1 then
            weight = weight * 3;
        end
    end
end Function
```

2.4 PESTN

PESTN (presented in Algorithm 4) is based on PEST algorithm presented in Sect. 2.3. The main difference compared to the PEST algorithm is that a threshold is `PASSED` if

the user answers correctly `rightAnswersToCertify` times at the threshold to be certified. The number of answers given at threshold N is saved into a vector `answers[]` at position N-1. Initially, the algorithm follows the PEST flow, until the set of certifiable thresholds is reduced to two consecutive levels. A `RIGHT` answer increments the number of right answers to the current threshold, a `WRONG` answer decrements the corresponding value. The test is `PASSED` if the user gives `rightAnswersToCertify` right answers at threshold `i`. In the case of two wrong answers at threshold `i`, the threshold is incremented until a higher threshold is certified or the threshold reaches the maximum certifiable. If the user does not answer correctly `rightAnswersToCertify` times at the same threshold, the test finishes in `FAILED` state.

3 Simulation protocol

In this paper, we do not apply the algorithms to a specific case study, but we run simulation assuming that we have a test and we want to certify a psychophysical threshold. The test starts at level N, it is decremented if the user guesses the answer, otherwise the level is incremented. If the user is not able to guess any answer the test fails. The choice of the next level at each step follows one of the algorithms described in Sect. 2. To test the operation of the algorithms, we have executed the tests on virtual patients, automatically generated with software. We have simulated 48000 patients and using the proposed algorithms we tried to certify different level of the psychophysical threshold under certification. For each user, we randomly select the answer (`RIGHT` or `WRONG`). We have preferred `RIGHT` answers when the patient is at level i and his psychophysical threshold is greater or equal to i, `WRONG` answers when the level i of the test is more difficult compared to his psychophysical threshold. To decide the distribution of `RIGHT` and `WRONG` answer, we have simulated four scenarios by assigning a probability to the `RIGHT` and `WRONG` answers as shown in Table 1.

The first scenario, *Scenario 0*, we have assumed that the user gives the `RIGHT` answer if he has the current psychophysical threshold, otherwise, the answer is `WRONG`. This is the ideal scenario, but it does not happen in practice because the user may choose a different answer e.g. because it tries to guess. We have considered these cases by adding two scenarios: *Scenario 1* and *Scenario 2*. The difference is in the probability of giving the `WRONG` answer. The `WRONG` answer is selected with a probability of 0.9 in *Scenario 1* and 0.75 in *Scenario 2* when the user does not have the current psychophysical threshold. The `RIGHT` answer is selected with a probability of 0.9 if the user has the current psychophysical threshold. *Scenario 2* is likely to happen when the user has a limited set of answers, for instance four, and a randomly chosen answer has a not negligible possibility to be the right one even if the current psychophysical threshold is below his psychophysical level. The last scenario starts from *Scenario 1* and it considers the decreasing of the level of attention. During the screening performed to measure the stereoacuity presented in [3], we have noticed that after around 8 steps of the test, the users were annoying and their level of attention decreased. To simulate this situation, after 8 steps, we have increased the probability of giving a `WRONG` answer.

Table 1. Probabilities of RIGHT and WRONG answers

	S0	**S1**	**S2**	**S3**	
Prob. RIGHT answer: currentLevel $\geq$ user psychophysical threshold	1	0.9	0.9	0.9	
Prob. WRONG answer: currentLevel < user psychophysical threshold or no psychophysical stimulus detection	1	0.9	0.75	0.9	(for the first 8 steps)
				0.75	(after 8 steps)

We have simulated all the algorithms with the same level of probabilities twice, in order to perform a *test-retest* assessment too. The goal is to evaluate test repeatability: the proposed algorithms guarantee the same level of certification in both simulations.

We have performed null hypothesis significance testing (NHST) for the evaluation and comparison of algorithms. NHST is a method of statistical inference by which an experimental factor is tested against a hypothesis of no effect or no relationship based on a given observation. In our case, we will formulate the null hypothesis following the schema that the algorithm X is no better than the others by considering the feature Y. Then, we will use the observations in order to estimate the probability or *p-value* that the null hypothesis is true, i.e. that the effect of X over the value Y is not statistically significant. If the probability is very small (below a given threshold), then the null hypothesis can be rejected.

4 Analysis of the Results

After we have gathered all the data form the simulation, we have performed a statistical analysis by answering a sequence of research questions (RQs) in order to extract useful information. For each RQ, we have formulated a null hypothesis (H_0) which posits the opposite compared to what we expect.

RQ1: Which is the algorithm that minimizes the number of false positive/false negative?

Besides measuring the psychophysical threshold of each person, the algorithm checks if that person has that psychophysical capability or not. The user may guess the right answer by chance, even if he or she is not actually capable of passing the test. On the other hand, we do not exclude that the patient gives the wrong answer even he or she has the desired capability. For these reasons, a test result could be *PASSED* when the patient does not have the capability, or *FAILED* even if the patient has the capability. These cases are called *false positive* and *false negative*. False positive is an error in the final result in which the test indicates the presence of the desired capability when in reality it is not present. Contrariwise, false negative is an error in which the test indicates the absence of the capability when the patient has it. We expect that one of the proposed algorithms minimize the number of false positive and false negative compared to the others.

To measure if an algorithm is better than the others in terms of false positive/false negative, we have introduced a statistical test called *Proportion Hypothesis Tests for*

Table 2. Proportion Hypothesis Tests for Binary Data: p-value

	S1	**S2**	**S3**
p-value FN	2.2e-16	2.2e-16	2.2e-16
p-value FP	1.58e-5	0.0001	0.0099

Table 3. Number of false positive and false negative

Algorithm	**False negatives**			**False positives**		
	S1	**S2**	**S3**	**S1**	**S2**	**S3**
STRICTN	2	35	1	56	74	77
BESTN	16	77	15	101	103	91
PEST	70	178	73	55	57	57
PESTN	9	79	8	52	54	58

Binary Data [5]. The result of this test is the p-value, based on this value we have decided to reject/accept the null hypothesis. The p-value threshold chose to determine if the null hypothesis is accepted or not is 0.005, this value guarantees that the obtained results are statistically significant. We started from two null hypothesis, one for the false positive and the other for the false negative:

H0_FP : No algorithm is better than other in false positive minimization.
H0_FN : No algorithm is better than other in false negative minimization.

The p-values obtained are shown in Table 2, the p-value of Scenario 0 is not reported because this is ideal scenario in which no false positive/false negative are detected. Given the results we can reject both the H0_FN and H0_FP. This means that there is an algorithm which guarantees a lower rate of false negative and false positive compared to the others.

Furthermore, this is confirmed by the number of false positives and false negatives detected as reported in Table 3. The data proves that STRICTN guarantees a lower rate of false negatives, followed by the PESTN. Regarding the false positive, PEST and PESTN perform similarly well, while STRICTN suffers in the scenarios S2 and S3. BESTN instead while produces an acceptable number of false negatives, it suffers from an excess of false positives.

Furthermore, to compare the algorithms we measure the *sensitivity* and the *specificity*. The sensitivity of a test is also called the true positive rate (TPR) and is computed as TP/P where TP is the number of true positive and P is the number of positive results. The sensitivity is the probability that a person without the capability (positive) reaches *FAILED* result. The specificity of a test, also referred to as the true negative rate (TNR), is computed as TN/N where TN is the number of true negative and N is the number of negative results. The specificity is the probability that a person with the capability reaches *PASSED* result. The values are reported in Table 4. In our test we assume that the sensitivity is more important than the specificity, since we want to be minimized the cases where the capability is not present but the test is PASSED nonetheless.

Table 4. Sensitivity and Specificity

Algorithm	Specificity				Sensitivity			
	S0	**S1**	**S2**	**S3**	**S0**	**S1**	**S2**	**S3**
STRICTN	1	.8231	.8206	.8086	1	.9998	.9948	.9993
BESTN	1	.7707	.7375	.7827	1	.9982	.9864	.9984
PEST	1	.8098	.7786	.9072	1	.9875	.9738	.9894
PESTN	1	.8216	.8384	.8245	1	.9984	.9884	.9978

Scenario S0 has the highest value of sensitivity and specificity (as expected) because it simulates the ideal situation in which all the patients have been certified with the target level and the patients without the capability have not been certified by the test. Since in terms of false negative there is an algorithm better than the other, we can notice that the sensitivity has different values based on the algorithm used and the scenario tested. The lowest value of sensitivity belongs to PEST algorithm in all scenarios, particularly in *Scenario 2*, while the algorithm with the highest value of sensitivity is STRICTN. The PESTN, although cannot perform well as the STRICTN, is very close to it. Regarding the specificity, PEST over performs the others, while BESTN is the worst.

RQ2: Which is the algorithm that minimizes the number of steps?

We have formulated the following null hypothesis to answer to RQ2:

H0 : All the algorithms perform test with the same number of steps.

First of all, we have computed the average of steps for each algorithm and for each scenario (see Table 5). From the table, we can observe that STRICTN and BESTN algorithms are those with the higher average number of steps, PEST is the algorithm with the lowest number of steps, while PESTN is in the middle. This can be confirmed by the Wilcoxon test [11]. We compare all the algorithms (in twos) to prove if one algorithm performs the test with fewer steps than the others. The p-value of Wilcoxon test are in Table 6, they have been computed under the hypothesis that the algorithm in the row takes fewer steps than the algorithm in the column. If p-value is less than threshold t=*0.005*, the null hypothesis is disproved otherwise it is approved. In some cases, e.g. the PEST columns, the p-values are higher than the threshold t, and we can disprove the null hypothesis *H0*.

RQ3: Which is the algorithm that guarantees a greater number of times in which the measured threshold is equal to target threshold?

We start from the following null hypothesis.

H0 : All the algorithms guarantee that the measured threshold is always not equal to the target threshold.

Table 5. The average of steps number

Algorithm	S0	S1	S2	S3
STRICTN	11.1	12.2	12.4	12.3
BESTN	10.2	10.8	11.0	10.9
PEST	4.61	4.73	4.80	4.76
PESTN	6.54	7.93	8.15	7.90

Table 6. Wilcoxon test for number of steps comparison: p-value

Algorithm	STRICTN	BESTN	PEST	PESTN
STRICTN	-	1	1	1
BESTN	<2.2e-16 (BESTN < STRICTN)	-	1	1
PEST	<2.2e-16 (PEST < STRICTN)	<2.2e-16 (PEST < BESTN)	-	<2.2e-16 (PEST < PESTN)
PESTN	<2.2e-16 (PESTN < STRICTN)	<2.2e-16 (PESTN < BESTN)	1	-

Given the target threshold (the one to be verified) and the measured threshold, we have computed the difference between them. Then, we have counted how many times they are different, the results are in Table 7.

All the algorithms in *S0* correctly certify the target value. In the other scenarios, since wrong answers are possible, sometimes the measured level is not equal to the target, especially for PEST algorithm. The more performing algorithm is the STRICTN because it runs sequentially all levels until the target is reached and it is required to guess three times the correct answer at the target level. The BESTN algorithm is slightly less performing than STRICTN.

Table 7. Times when measured threshold is equal to target threshold over 5,604 *PASSED* simulations - the *FAILED* tests are excluded

	S0	S1	S2	S3
STRICTN	5,604	4,956	4,929	4,943
BESTN	5,604	4,466	4,608	4,582
PEST	5,604	2,155	2,223	2,123
PESTN	5,604	4,632	4,624	4,582

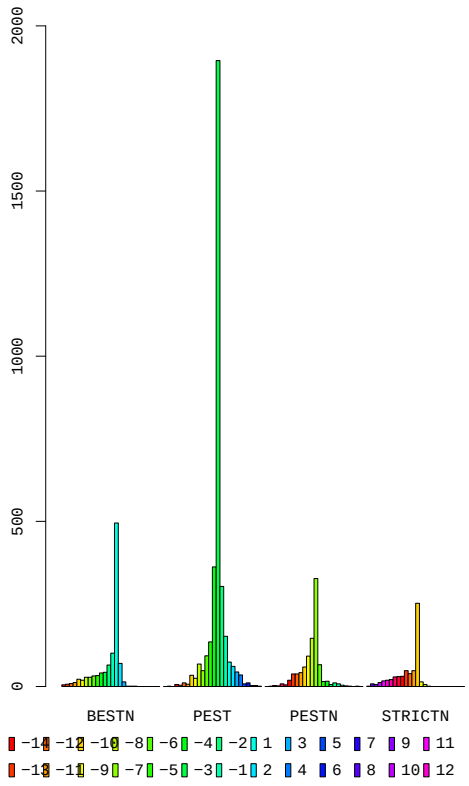

Fig. 1. Difference between target level and measured level in Scenario 1

RQ4: Which is the algorithm with the minimum difference between target level and measured level?

When the difference between the target level and measured level is not equal to zero, we are interested to know this value. To answers at this RQ, we start from the following null hypothesis:

H0 : The difference between the target level and measured level is the same regardless of the algorithm.

The difference between target level and measured level is shown in Fig. 1, Fig. 2 and Fig. 3. In *Scenario 1*, the percentage of cases in which target and measured level are different for each algorithm simulation is the following: 27,00% PEST, 13,31% PESTN, 8,30% STRICTN (the percentage is computed over the 20.000 simulations for

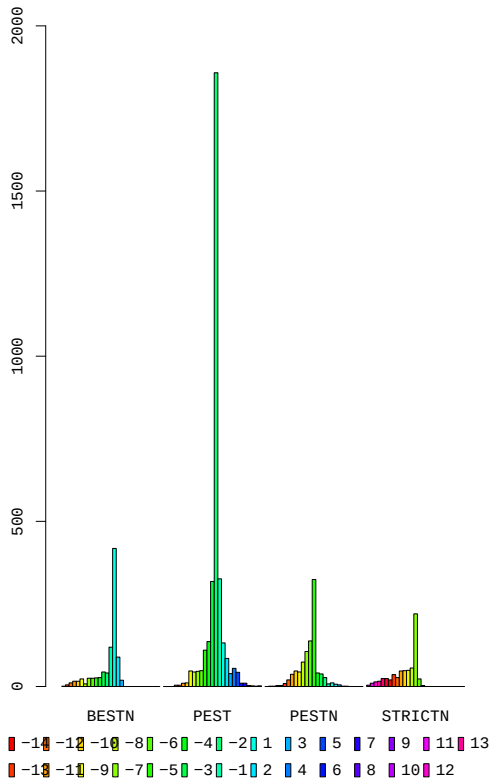

Fig. 2. Difference between target level and measured level in Scenario 2

each algorithm). We have further investigated for each algorithm the difference between target and measured level. PEST algorithm certifies user with one level plus or minus in 48,01% of cases and two levels plus or minus in 23,93% of cases. The difference between target and measured level is more than two levels in 28,06% of cases. While these two algorithms have a distribution centered on ± 1 and ± 2, PESTN and STRICTN distributions are centered on [-4,-1]. STRICTN certifies most of the tests (54,22%) with one level minus and 11,45% of them are certified with two levels minus. Furthermore, we have noticed that all the algorithms, except PEST, are "pessimists" because when the target and measured level are different, in many cases, they certify a higher level compared to the target. With the introduction of higher error probability, *Scenario 2*, the percentage of cases in which target and measured level are different for each algorithm simulation is the following: 42,25% PEST, 22,72% PESTN, 13,33% STRICTN

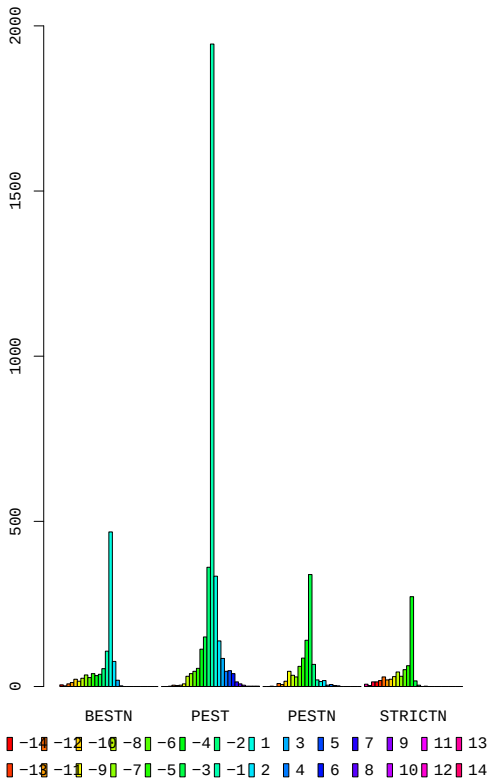

Fig. 3. Difference between target level and measured level in Scenario 3

(the percentage is computed over the 20.000 simulations for each algorithm). As expected, the percentages are higher compared to *Scenario 1* because the probability of the wrong answer has been incremented. The difference between target and measured level is centered on [-1,3] for PEST algorithms and [-2,2] for PESTN and STRICTN algorithms. In details, PEST has 68,30% of cases in the [-1,3] interval, while the percentage in interval [-2,2] is 70,47% and 72,47%for PESTN and STRICTN respectively. S3 has values similar to S1, in some cases the difference is a little bit higher.

RQ5: Which is the best algorithm with the best performance in test-retest?

Test-retest evaluates the repeatability of a test administered at two different times, T1 and T2. A test is repeatable if the measure does not change between the two measurements, under the hypothesis that in T1 and T2 the symptomatology is not changed.

We have started the analysis from the following null hypothesis:

H0 : All the algorithms have the same performance in test-retest.

We have applied the Pearson Correlation Coefficient to measure the reliability of test-retest. The results are shown in Table 8. As expected, in *Scenario 0* the correlation is equal to 1 for all the algorithms because this scenario guarantees that for every simulation the certified level is always the target. In the other scenarios, the algorithm with the highest correlation is PESTN, which has good reliability coefficients (from 0.87 to 0.9). At the opposite, the algorithm with the lower correlation is PEST which has poor repeatability. Similarly to PESTN, BESTN and STRICTN have good reliability in all the scenarios.

Table 8. Pearson correlation test-retest

	S0	**S1**	**S2**	**S3**
STRICTN	1	0.88	0.85	0.85
BESTN	1	0.84	0.86	0.86
PEST	1	0.81	0.77	0.81
PESTN	1	0.90	0.87	0.88

5 Discussion

In the previous section, we have answered to a set of RQs to measure sensitivity and sensibility, number of steps, number of times that the measured level is equal to the target level, the difference between target level and measured level (when they are different), and the test-retest reliability. In this section, we want to discuss the results and provide some guidelines to choose the algorithm based on the test goal. For each RQ we have assigned a score from one to three (see Table 9), one is assigned to the algorithm which better satisfies the research question, three is assigned to the worst algorithm under analysis.

Table 9. Comparison between RQs: which algorithm guarantee the best performance?

Algorithm	**RQ1**	**RQ2**	**RQ3**	**RQ4**	**RQ5**
STRICTN	1	4	1	1	2
BESTN	3	3	4	3	3
PEST	4	1	3	4	4
PESTN	2	2	2	2	1

STRICTN is the algorithm with best performance in guaranteeing the lowest number of false positive and false negative, target level, and measured level are the same

most of the time and when they are different the difference is mostly ± 1 level. The algorithm with worst performance is PEST. Nevertheless, the number of steps required for the test is very low, in most cases the target level is not equal to the measured level, and the test-retest reliability is the lowest. BESTN algorithm has similar performance to PEST, moreover it requires lot of steps to perform tests. When it is required an algorithm with good performance, but with a limited number of steps to complete the test, PESTN is a good compromise because it can be applied in around half of the steps compared to STRICTN. It has high sensitivity and sensibility, and the measured level is equal to the target in a large number of cases and when they are not equal the difference is minimal. Furthermore, it guarantees the best test-retest reliability.

Generally, we did not notice a big difference between *S1* and S3, in which we have included the decreasing of attention. This can be because we have considered tests with limited number of steps. We think that this scenario requires a more in-depth analysis, because, from our experience, when tests become long the attention (and also the patience) of patients decrease.

6 Related work

In this section, we present the algorithms used in literature for the stereoacuity measurement. In papers [1,8] the authors apply the PEST algorithm to measure stereoacuity using the Freiburg Test and, as demonstrated also by our case study, the proposed algorithm allowed to save time during the stereoacuity measurement. We found that Staircase algorithm is often used in the literature, with some minimal differences. In papers [17,10,16,15], stereoacuity is measured using staircase, the disparity is increased/-decreased of one level. The disparity is increased of one level and decreased of two levels in paper [6]. In paper [14], staircase is compared to book based clinical testing and the result is that the threshold measured with digital test is more reliable also due to the possibility to increase the number of levels of disparity.

A comparison among algorithms is presented in [9]. Also in that paper, the results show that faster converging maximum-likelihood procedures like PEST and BESTN offer some benefit over longer staircase procedures. especially when testing must be accomplished very quickly, as in testing animals or infants.

The necessity to devise new algorithm for testing infants and children is advocated in [7], where the authors consider how best to maximize speed, accuracy, and reliability.

7 Conclusion

In this paper we have presented four algorithms for the evaluation of psychophysical thresholds. The tests are performed on virtual patients, because it was very difficult to find a lot of patients available to perform a test many times. In the future we plan to define a protocol to test the algorithms on real patients, for example to measure the stereoacuity using our mobile application [4]. Our simulations show that the choice of the best algorithm to follow, depend on many factors. Algorithm like STRICTN performs generally well, except in cases in which the test cannot last so many steps. In those cases, other algorithms like our PESTN are more efficient.

References

1. Bach, M., Schmitt, C., Kromeier, M., Kommerell, G.: The freiburg stereoacuity test: automatic measurement of stereo threshold. Graefe's Archive for Clinical and Experimental Ophthalmology **239**(8), 562–566 (aug 2001). https://doi.org/10.1007/s004170100317
2. Bonfanti, S., Gargantini, A.: Comparison of algorithms to measure a psychophysical threshold using digital applications: The stereoacuity case study. In: Pesquita, C., Fred, A.L.N., Gamboa, H. (eds.) Proceedings of the 14th International Joint Conference on Biomedical Engineering Systems and Technologies, BIOSTEC 2021, Volume 5: HEALTHINF, Online Streaming, February 11-13, 2021. pp. 78–86. SCITEPRESS (2021). https://doi.org/10.5220/0010204000780086, https://doi.org/10.5220/0010204000780086
3. Bonfanti, S., Gargantini, A., Esposito, G., Facchin, A., Maffioletti, M., Maffioletti, S.: Evaluation of stereoacuity with a digital mobile application **259**(9), 2843–2848. https://doi.org/10.1007/s00417-021-05195-z, https://doi.org/10.1007/s00417-021-05195-z
4. Bonfanti, S., Gargantini, A., Vitali, A.: A mobile application for the stereoacuity test. In: Duffy, V.G. (ed.) Digital Human Modeling. Applications in Health, Safety, Ergonomics and Risk Management: Ergonomics and Health. pp. 315–326. Springer International Publishing, Cham (2015)
5. Fleiss, J.L., Levin, B., Paik, M.C.: Statistical methods for rates and proportions; 3rd ed. Wiley Series in Probability and Statistics, Wiley, Hoboken, NJ (2003)
6. Hess, R.F., Ding, R., Clavagnier, S., Liu, C., Guo, C., Viner, C., Barrett, B.T., Radia, K., Zhou, J.: A robust and reliable test to measure stereopsis in the clinic. Investigative Opthalmology & Visual Science **57**(3), 798 (mar 2016). https://doi.org/10.1167/iovs.15-18690
7. Jones, P.R., Kalwarowsky, S., Braddick, O.J., Atkinson, J., Nardini, M.: Optimizing the rapid measurement of detection thresholds in infants. Journal of Vision **15**(11), 2 (aug 2015). https://doi.org/10.1167/15.11.2
8. Kromeier, M., Schmitt, C., Bach, M., Kommerell, G.: Stereoacuity versus fixation disparity as indicators for vergence accuracy under prismatic stress. Ophthalmic and Physiological Optics **23**(1), 43–49 (jan 2003). https://doi.org/10.1046/j.1475-1313.2003.00089.x
9. Leek, M.R.: Adaptive procedures in psychophysical research. Perception & Psychophysics **63**(8), 1279–1292 (nov 2001). https://doi.org/10.3758/BF03194543
10. Li, R.W., So, K., Wu, T.H., Craven, A.P., Tran, T.T., Gustafson, K.M., Levi, D.M.: Monocular blur alters the tuning characteristics of stereopsis for spatial frequency and size. Royal Society Open Science **3**(9), 160273 (sep 2016). https://doi.org/10.1098/rsos.160273
11. Noether, G.E.: Introduction to Wilcoxon (1945) Individual Comparisons by Ranking Methods, pp. 191–195. Springer New York, New York, NY (1992)
12. Stevens, S.S.: Problems and methods of psychophysics. Psychological Bulletin **55**(4), 177–196 (1958)
13. Taylor, M.M., Creelman, C.D.: Pest: Efficient estimates on probability functions. The Journal of the Acoustical Society of America **41**(4A), 782–787 (1967)
14. Tidbury, L.P., O'Connor, A.R., Wuerger, S.M.: The effect of induced fusional demand on static and dynamic stereoacuity thresholds: the digital synoptophore. BMC Ophthalmology **19**(1) (jan 2019). https://doi.org/10.1186/s12886-018-1000-2
15. Ushaw, G., Sharp, C., Hugill, J., Rafiq, S., Black, C., Casanova, T., Vancleef, K., Read, J., Morgan, G.: Analysis of soft data for mass provision of stereoacuity testing through a serious game for health. In: Proceedings of the 2017 International Conference on Digital Health - DH '17. ACM Press (2017). https://doi.org/10.1145/3079452.3079496

16. Vancleef, K., Read, J.C.A., Herbert, W., Goodship, N., Woodhouse, M., Serrano-Pedraza, I.: Two choices good, four choices better: For measuring stereoacuity in children, a four-alternative forced-choice paradigm is more efficient than two. PLOS ONE **13**(7), e0201366 (jul 2018). https://doi.org/10.1371/journal.pone.0201366
17. Wong, B.P.H., Woods, R.L., Peli, E.: Stereoacuity at distance and near. Optometry and Vision Science **79**(12), 771–778 (dec 2002). https://doi.org/10.1097/00006324-200212000-00009
18. Zhang, P., Zhao, Y., Dosher, B.A., Lu, Z.L.: Evaluating the performance of the staircase and quick change detection methods in measuring perceptual learning. Journal of Vision **19**(7), 14 (jul 2019). https://doi.org/10.1167/19.7.14