CCDC DAC-TR-2020-045
July 2020

Limiting Runs for Computing Probability Estimates from Computationally Intense Models

by Craig D. Andres

DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

DISCLAIMER

The findings in this report are not to be construed as an official Department of the Army position unless so specified by other official documentation.

WARNING

Information and data contained in this document are based on the input available at the time of preparation.

TRADE NAMES

The use of trade names in this report does not constitute an official endorsement or approval of the use of such commercial hardware or software. The report may not be cited for purposes of advertisement.

CCDC DAC-TR-2020-045
July 2020

Limiting Runs for Computing Probability Estimates from Computationally Intense Models

by Craig D. Andres
CCDC Data & Analysis Center

DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED *(From - To)*
July 2020	Technical Report	1 January 2019–16 April 2020

4. TITLE AND SUBTITLE
Limiting Runs for Computing Probability Estimates from Computationally Intense Models

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Craig D. Andres

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Director
U.S. Army CCDC Data & Analysis Center
6896 Mauchly Street
Aberdeen Proving Ground, MD

8. PERFORMING ORGANIZATION REPORT NUMBER

CCDC DAC-TR-2020-045

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Computing probability estimates in a complex model with stochastic logic has challenges with respect to the nature of the underlying distribution, which in our effort is assumed to be binomial. We use a highly complex and computationally intense model to estimate probabilities of multiple outcomes conditioned on engagement scenarios and using thousands or even millions of iterations. Because of the amount of computation time needed, and the increasing use of the model, limiting the number of iterations is important. From a binomial perspective we have a response range on (0,1), but our model response range includes the interval endpoints and thus is [0,1]. It is the endpoints of zero and one that provide those challenges. Using a fixed value as a requirement or a relative requirement is an oversimplified approach to a conditional problem. This report details a hybrid approach to provide a user-customizable solution.

15. SUBJECT TERMS
binomial confidence intervals, high-performance computing, probability models, logistic, Agresti-Coull interval, stopping rules

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Craig D. Andres
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAME AS REPORT	28	19b. TELEPHONE NUMBER *(include area code)* 410-278-6721

Table of Contents

List of Figures

List of Tables

Acknowledgements

This effort was motivated by Long Range Precision Fires and Next Generation Combat Vehicles cross-functional team efforts. A special thanks for technical feedback from U.S. Army Combat Capabilities Development Command (CCDC) Data & Analysis Center (DAC) colleagues James Edwards, Kinetic Experimentation Branch; Melissa Hawkins, Aviation and Air Defense Branch; and Timothy Myers, Warfighter Modeling and Simulation Branch. A special thanks for model implementation to Jim Hunt, CCDC Data & Analysis Center Joint Software Engineering Branch.

1. INTRODUCTION AND PURPOSE

Computing probability estimates in a complex model with stochastic logic has challenges with respect to the nature of the underlying distribution, which in our effort is assumed to be binomial. The Advanced Joint Effectiveness Model (AJEM) is a highly complex and computationally intense model used to estimate probabilities of multiple outcomes for vehicle/threat engagement scenarios using thousands or even millions of iterations. Because of the amount of computation time needed, limiting the number of iterations is important. As the usage of this model has grown, the natural questions of "when to stop?" and "how close of an estimate is good enough?" were asked with concern for conserving computational resources. The random variation built into the model code suggests a statistical approach. From a binomial perspective, we have a response range on (0,1), but our model response range is really [0,1]. In the normal distribution realm, we could set a constant radial requirement so that once we are within R units after N runs we are within a specified confidence interval. In the binomial realm, estimation when the true value is close to zero or one does not behave the same as when the true value is nearer to 0.5. There are issues with using a fixed value as a requirement or a relative requirement. If zero or one are the true values, then a hard stop would be needed as any confidence-based approach would never converge to zero or one. So a hybrid approach that can be customized by the user is outlined and detailed in this report.

2. METHODOLOGY

In AJEM, the conditional success with respect to a played out engagement scenario is defined by the user and relative to the perspective of lethality or survivability. This report is mostly written with respect to lethality so that it is easier to focus on the overall concept, but the use is always with respect to however mission success for a given engagement scenario is defined by the user. Our focus is estimating values of Pk, which indicates the probability of kill for a given scenario or P(kill | scenario). We want to be able to estimate Pk, or a set of Pk's based on different scenarios, within a defined interval based on confidence or prediction-based intervals, but within a reasonable number of AJEM iterations of the given scenario. The focus is on a single Pk estimate, though Pk values can come in sets. The algorithm can be designed to focus on a specific Pk estimate. If the predictions are multivariate (more than one scenario), the user can have a group requirement such as all Pk values must meet the stopping criteria, or even focus on the minimum or maximum of the estimated Pk's.

Our goal is to develop a stopping rule for convergence of probability estimate(s). The engagement scenarios are usually computationally complex, and take a lot of time to complete. We will need an input for a cap on the number of iterations completed, and we will use a default cap of 1000 iterations. We know that the range of probabilities for mission success for a given engagement scenario ranges from zero to one (i.e., $0 \leq p \leq 1$). The underlying distribution is assumed to be binomial with a constant p as the probability of success. Convergence becomes more challenging when that probability is nearer to 0 or 1, so we propose framing the stopping rules in such a way as to reduce the occurrence of reaching the capped number of iterations for those cases. So we will look at a three-part set of stopping rules to better manage the computational time needed. Note that this approach is symmetric to 0.5 so that near-zero and near-one behavior is identical. The three parts are:

1. A stopping rule for the inner part of [0,1] centered at 0.5 that is well defined statistically. The inner interval will be user defined [c,1 – c] as illustrated in Figure 1.
2. A rule for the outer parts of the same interval that results in an adequate estimate without breaking the defined cap.
3. A user input for a hard cap (default will be 1000 iterations).

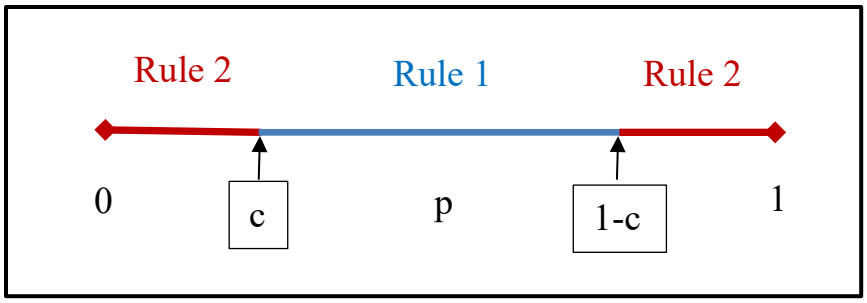

Figure 1. Stopping rule intervals on 0 ≤ p ≤ 1 where 0 < c < 0.5

Overall, this is a piecewise practical approach giving the user customizable control in limiting the number of model runs. Rule 1 addresses our ideal scenario of having a constant radial requirement (within R units from the truth) for most of [0,1]. Rule 2 addresses the difficulty near zero and one using a constant relative requirement (within R/p of the estimate, for the near-zero case). Rule 3 addresses the zero and one case (or so close to zero or one it does not matter, practically speaking) so that we do not get caught in an infinite loop of iterations. Results for our AJEM engagement scenarios are used in higher level models, so making this algorithm customizable for the needs of those users is important. For example, such a user may assign importance to a probability of 0.01, where another user may want that estimate equated to zero. Note that this is an active algorithm that depends on the estimate for p in the last run.

The stopping rules are based on user inputs, the number of iterations completed, and a mathematical formula or two that can satisfy the three-rule approach. To find that formula, the concepts behind confidence intervals for the expected value of probability of success using a binomial distribution are used to construct a discriminator, or score, to control the number of iterations of the engagement scenario that are run in AJEM. Four approaches using a confidence interval formula were tested in simulation. These scores tracked each other very closely, and were nearly identical after 50 or so iterations. All of these methods use a version of continuity correction for the probability estimate to adjust to a normal approximation for the interval width.

Based on some experimentation, the Agresti-Coull interval appeared more consistent for lower values of n, and it has a relatively simple formula, where n is the number of trials. For our case, n is the number of iterations that will be used for the engagement scenario. The z value is the $1 - \alpha/2$ quantile of the standard normal distribution. The actual count of successes out of the n iterations is X.

$$p = p_k \qquad \text{true probability of mission success for a given scenario,} \tag{1}$$

$$\bar{p} = \frac{X}{n} \qquad \text{the expected value and familiar estimate for } p \text{ (not used)} \tag{2}$$

$$\tilde{n} = n + z^2 \qquad \text{an adjusted number of trials} \tag{3}$$

$\tilde{p} = \frac{1}{\tilde{n}}\left(X + \frac{z^2}{2}\right)$ a continuity corrected estimate for p_k (4)

Note that when $n = 0$, $\tilde{p} = 0.5$, so we avoid any division by zero and always have an initial estimate. Before even one trial the estimate gives $\tilde{p} = 1/2$ using $z = 2$ (X = 0 of course). After 10 trials if X = 0 or 10, $\tilde{p} = 1/7$ or 6/7, respectively. We see that $\tilde{p}$ anchors the estimate away from the endpoints relative to the number of trials. In the algorithm, we could start the estimate at
$n = 0$, but we will set a minimum set of runs defined by the user, with the default set at 10, to get a more accurate initial estimate. Using these adjustments, the Agresti-Coull confidence interval for p is given by

$$\tilde{p} \pm z\sqrt{\frac{\tilde{p}}{\tilde{n}}(1-\tilde{p})} \tag{5}$$

So the interval radius from Agresti-Coull will be used as our "score" value and is used as the basis for determining our desired proximity to the estimate.

$$\text{AC.Score} = z\sqrt{\frac{\tilde{p}}{\tilde{n}}(1-\tilde{p})} \tag{6}$$

Let *dne* represent the distance from the nearest endpoint to the true value of p (i.e., *dne = min [p, 1 – p]*), which can be estimated with any estimate of p. This value will be checked against a value related to c (see Figure 1) to determine which stopping rule applies.

$$dne = \min(\tilde{p}, 1-\tilde{p}) = 0.5 - |p - 0.5| \tag{7}$$

So if $\tilde{p}$ is 0.5, then *dne* = 0.5, if p = 0.3 or 0.7, then *dne* = 0.3. One approach will use the ratio of estimates for *dne* and *AC.Score*,

$$\text{AC.Ratio} = \frac{AC.Score}{dne} = \frac{z}{dne}\sqrt{\frac{\tilde{p}}{\tilde{n}}(1-\tilde{p})} \tag{8}$$

Note that the algebraic structure of the score and ratio equations allows it to be solved for n. We can use estimates for p or hypothesized values. Equations 9 and 10 replace *AC.Score* and *AC.Ratio* with either a user-defined *Input.Radius* or *Input.Ratio*, respectively. These equations can be used to find the approximate number of iterations needed to get either desired result.

$$\tilde{n} = \frac{z^2\tilde{p}(1-\tilde{p})}{Input.Radius^2} \tag{9}$$

or

$$\tilde{n} = \frac{z^2\tilde{p}(1-\tilde{p})}{dne^2\ Input.Ratio^2} \tag{10}$$

Then using Equation 3 our number of iterations is

$$n \cong round(\tilde{n} - z^2) \tag{11}$$

These are not used in the applied algorithm, but are neighborhood estimates of n that could be used by a user to consider how input values can be engineered to stay within an affordable amount of computation time. In practice, the algorithm designed will be checking the rules at each iteration, meaning that it will not solve for *n* directly. The two approaches, a fixed interval rule versus a fixed ratio rule, have different dynamics statistically. To understand how these calculations work, suppose $z = 1.96$ for a two-sided 95% confidence interval, p is 0.5, and we want the radius of our interval to be within 20% of the estimate (so 0.5 ± 0.1). Then n would be about 88 iterations. Increasing the input ratio to 30% would require n to be about 39 iterations. The further the true p is from 0.5, the lower n will need to be to get within a fixed interval radius, which is counterintuitive to the expected behavior of the probability estimate. We will see that effect in our results (Table 1). In this format initial estimate of p used cannot be zero or one.

Table 1. Number of Iterations Needed Based on Radius vs. Ratio for Rule 1

p	z (2-tail Confidence)	Input Radius	Iterations Needed Using Score	Input Ratio (Radius[a])	Iterations Needed uUing Ratio
0.1 or 0.9	1.96 (95%)	0.05	134	20% (0.02)	860
0.3 or 0.7	1.96 (95%)	0.05	319	20% (0.06)	220
0.5	1.96 (95%)	0.05	380	20% (0.10)	92
0.1 or 0.9	1.96 (95%)	0.10	31	30% (0.03)	380
0.3 or 0.7	1.96 (95%)	0.10	77	30% (0.09)	96
0.5	1.96 (95%)	0.10	92	30% (0.15)	39
0.1 or 0.9	1.645 (90%)	0.10	22	30% (0.03)	268
0.3 or 0.7	1.645 (90%)	0.10	54	30% (0.09)	67
0.5	1.645 (90%)	0.10	65	30% (0.15)	27
0.1 or 0.9	1.28 (80%)	0.10	13	30% (0.03)	163
0.3 or 0.7	1.28 (80%)	0.10	33	30% (0.09)	41
0.5	1.28 (80%)	0.10	39	30% (0.15)	17

[a]The (radius) in the fifth column parenthesis is the corresponding radius to the ratio.

Using the AC.Ratio comes with additional complexities. With it we are assessing using a ratio of a probability, which sounds like a percentage of a percentage, which can be confusing and we have tried to avoid referring to it in those terms. The choice of q could be any ratio between zero and one, where closer to one will require fewer runs. The user must make a choice for q that corresponds to mission needs for accuracy at the lower end. From a mission view it makes sense to have a fixed radius, but even $p = 0.5 \pm 0.1$ and $p = 0.2 \pm 0.1$ can be very different with respect to practical interpretation. The dne value changes with every new iteration, as it is based on the estimate for p.

The results in Table 1 use the formulas in Equations 9, 10, and 11. Note that using the fixed radius or using the ratio behaves differently in terms of decreasing and increasing needs for sample size relative to the distance from the end points. Columns 3 and 4 are independent of Columns 5 and 6. The radius is included. When keeping the radius at a fixed value, it is easier for smaller values of p to result in convergence because the variance $p(1 - p)$, which drives the sample size calculation, is lower. But this fixed value makes much less sense as p nears 0 or 1, since it results in limits outside of [0,1] and there is no resolution on values that are close to 0 or 1. To illustrate that, if we select a radius of 0.1, and our true p is 0.02, then our lower limit (LL) would estimate –0.08 (a negative probability is of course impossible) and our upper limit (UL) would be 0.12, which is six times the estimate, which could be a bad choice if a conservative estimate is needed.

It may be that something in-between these two approaches could make better sense mathematically, but would not likely make much sense from a practical sense in how it would be interpreted. Remember we are building a default case with default input, and different rules could be applied within this framework by the user.

What we get out of Table 1 is the sense that the Rule 1 and Rule 2 approaches have opposite associations for the p to n relationship relative to the distance from the center of (0,1). This is driven by the variance component from the confidence interval that is part of *AC.Score*. The algorithm parameters can be set so that it only uses Rule 1 (using $c = 0$) or Rule 2 (using $c = 0.5$) with the hard stop for Rule 3.

Using the score to fix an interval length is a familiar approach as it uses a fixed width target for the radius of the confidence interval for the estimate of p with respect to Rule 1. Transitioning to Rule 2 if the estimate is less than c, or greater than $1 - c$, we switch to use the ratio to define the confidence interval so that the estimate is within a defined ratio of the distance to the nearest end (q**dne*). Our chosen default value for q is 0.5, and this can be changed by the user in the application. Using the default value while subject to Rule 2, we continue iterations of the model until confidence interval radius of our estimates is within half of the value of the *dne*. This is further illustrated in Figure 2.

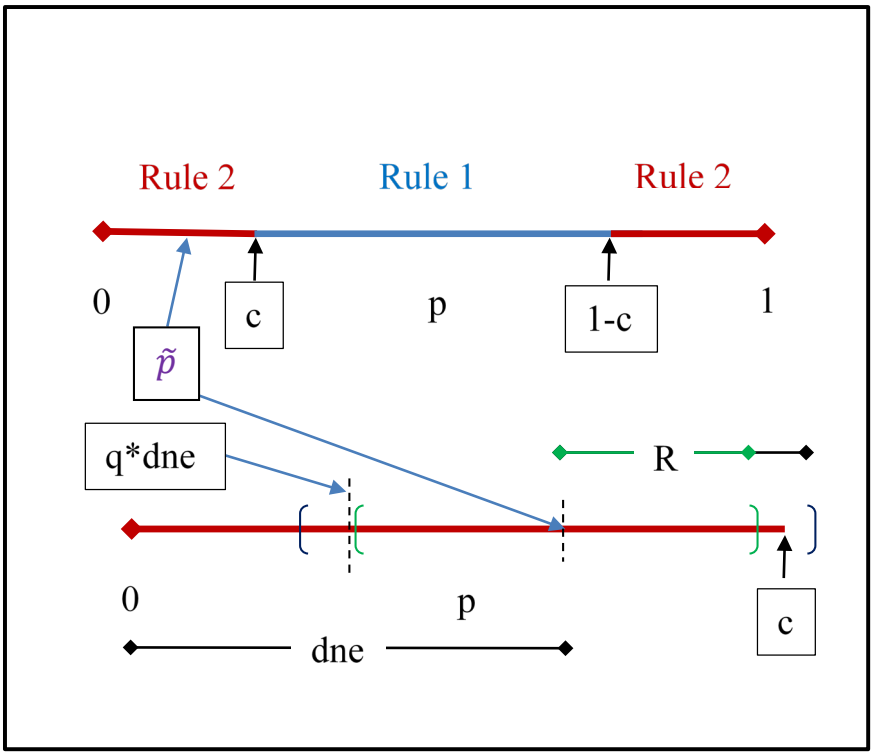

Figure 2. Rule application demonstration

In Figure 2, the application of Rule 2 is illustrated where p is estimated at $\tilde{p}$ at less than c. The dne is shown at the bottom, and in this zone is the same as $\tilde{p}$. The ratio q is illustrated as 0.5 and q^*dne is shown as the dotted vertical line, which defines our requirement. The radius of the confidence interval at any given iteration is notated as R for two example outcomes in green and black. The algorithm will stop the runs if the left side of the confidence interval falls on the $\tilde{p}$ side of q^*dne. Where R/left-bracket is black and to the left of q^*dne, the algorithm will continue until it reaches approximately where the green R/left-bracket is within q^*dne of $\tilde{p}$. If $\tilde{p} > 1 - c$, then this scenario would be mirrored around $1 - \tilde{p}$.

In a nutshell, this algorithm applies Rule 1 using the fixed interval (r = constant) approach when $c < \tilde{p} < 1 - c$ and Rule 2 otherwise for the $\tilde{p}$ of the current iteration, so if the true p is near c or $1 - c$ it is likely that either rule could find the stopping condition. The user must decide the level of rarity of an event needs to be found with some resolution, and adjust the value of c, and the cap, to reflect that needed endpoint resolution (EPR).

Our defaults are not meant to become canon, but we use $c = 0.2$. The default fixed radius is set at $r = 0.1$ for Rule 1, and the ratio for Rule 2 is set at $q = 0.5$, with a hard count stop as Rule 3 defaulted to 1000 runs. Note that at the transition point c, the interval radius is the same for both Rule 1 and Rule 2 (i.e., $r = 0.1 = 0.5*0.2 = q^*c$), which allows for a smooth transition between the rules. Though not required, it is the sort of transition that keeps mathematicians happy.

3. ALGORITHM

Defaults (can be changed by user):

Confidence level = 90%

Rule 2 ratio q = 0.5

r = 0.1

$c = r/q$

Initial runs for p estimate =10

Cap on iterations = 1000

Stop runs and calculate/report iteration count, estimate, interval limits, and confidence when one of the following conditionals is met:

1. If $q * dne > r$ then stop if AC.Score ≤ r
2. If $q * dne \leq r$ then stop if AC.Ratio ≤ q
3. or equivalently AC.Score≤ $q * dne$
4. If iterations ≥ cap

A color coding such as the example in Table 2 can be defined and applied to report tables and graphics for improved visualization and reporting as seen in Table 3.

Table 2. Color Code Example

Rule Result	Flag	Interpretation
1	Green	Confidence interval radius within specified constant for Rule 1
2	Blue	Confidence interval radius within specified ratio for Rule 2
3	Purple	Exceeded cap, user defines estimate as equivalent to zero or one

4. TEST RESULTS

Table 3 shows the performance across true values of p for the above algorithm with default values, with the exception that confidence is set at 95% instead of 90% to allow comparison to Table 1. Unlike Table 1 that used direct calculation, this set of results used an R simulation of the algorithm with a set seed value for all rows run with 10 initial runs to get the initial estimate for Pk.

The program then returns the number of runs completed once a stopping rule was triggered.

Table 3. Number of Iterations Needed Based on Algorithm at Defaults

True p	z (2-tail Confidence)	Input Radius	Iterations Needed	$\tilde{p}$	LL	UL
0.01	1.96 (95%)	0.1	719	0.022	0.011	0.033
0.05	1.96 (95%)	0.1	225	0.065	0.033	0.097
0.1	1.96 (95%)	0.1	140	0.097	0.048	0.145
0.2	1.96 (95%)	0.1	67	0.239	0.140	0.338
0.3	1.96 (95%)	0.1	74	0.282	0.182	0.382
0.4	1.96 (95%)	0.1	86	0.366	0.267	0.466
0.5	1.96 (95%)	0.1	93	0.505	0.406	0.605
0.6	1.96 (95%)	0.1	86	0.634	0.534	0.733
0.7	1.96 (95%)	0.1	74	0.718	0.618	0.818
0.8	1.96 (95%)	0.1	67	0.761	0.662	0.860
0.9	1.96 (95%)	0.1	140	0.903	0.855	0.952
0.95	1.96 (95%)	0.1	225	0.935	0.903	0.967
0.99	1.96 (95%)	0.1	719	0.978	0.967	0.989

Note that there is a symmetry of results by the true value of p; this is because the same seed value was used for random number generation on each scenario. None of these cases exceeded the cap of iterations (set at 1000), though additional runs at 0.01 exhibited Rule 3 cap usage most of the time. Note that the first and last scenario resulted in a confidence interval that did not contain the true value of p, thus confirming that bias can happen near zero and one depending on the parameters used to get the desired EPR. Endpoints could be further reduced by using a higher input ratio (q). Using $q = 0.75$ instead of $q = 0.5$ for $p = 0.99$ results in an estimate of 0.970 with LL at 0.948, and UL at 0.992 in 225 runs. Note also that the border values to the decision rules are $p = 0.2$ and 0.8, which do appear to complete within the scope of Rule 1. The algorithm was written to allow the decision for convergence on both Rule 1 and Rule 2 simultaneously. As noted, the rules could be written so that only one of the two rules is queried after the initial runs by setting the c parameter to 0 or 0.5. A graph illustrating how the resulting confidence intervals are based on Rules 1 and 2 is shown in Figure 3. It is a symmetric relationship about $p = 0.5$. Note the slight s-curve from 0.2 – 0.8, though it is nearly linear. This curvature becomes more pronounced if we use $p = 0.1$ and 0.9 as the border values.

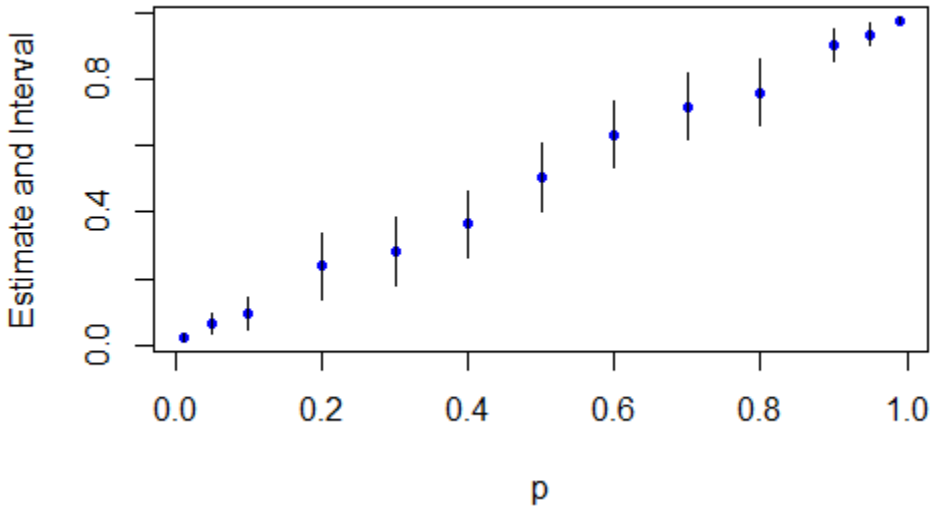

Figure 3. Estimates of p with confidence bounds by true value of p

5. R PROGRAM INPUTS AND CODE

The R code that follows differs from the algorithm only because of the input of the true value of p (Ptrue), which in application will be the unknown expected response value produced by AJEM. All other inputs are user-defined so that they can be customized to the needs of the user. This can of course be programmed in almost any computer language (Agresti and Coull, 1998).

Inputs

- *Ptrue* is the true value of mission success for the simulation. This is the value being estimated by the algorithm, and will not be in the actual AJEM code.
- *Seed1* is user-defined with default for random number generation.
- *Initialruns* is the user-defined number of initial iterations of the mission with default to get an initial estimate for ptrue.
- *ITcap* is the user-defined cap of maximum number of iterations to be performed.
- *r1* is the user-defined fixed interval length for Rule 1.
- *q1* is the user-defined ratio to define the target interval length with respect to Rule 2.
- *Confidence* is the user-defined confidence value for the two-sided interval.

```
ACsim=function(Ptrue=0.5,seed1=1324,initialruns=10,ITcap=1000,r1
=0.1,q1=0.5,Confidence=0.90){
  set.seed(seed1)

  c1=r1/q1
  z1=qnorm(1-(1-Confidence)/2)
  alpha=1-Confidence/2
  Successes=rbinom(1,initialruns,Ptrue)
  Phat=Successes/initialruns
  Ptilde=1/(initialruns+z1^2)*(Phat*initialruns+z1^2/2)
  runs1=initialruns

  Flag1=TRUE

  #first 10 for Pest
 while (Flag1){
   runs1=runs1+1
   Successes=Successes+rbinom(1,1,Ptrue)
   Phat=Successes/runs1
   Ptilde=1/(runs1+z1^2)*(Phat*runs1+z1^2/2)
   AC=z1*sqrt(Ptilde*(1-Ptilde)/(runs1+z1^2))
   dne=.5-abs(.5-Ptilde)

   #Rule3
   if (runs1>=ITcap) {
```

```
        LL=Ptilde-AC
        UL=Ptilde+AC
     #if Rule 3 is triggered, a ratio of interest would be
AC/dne
     #too illustrate how far different it is from the chosen q
     Flag1=FALSE
   }
   #Rule 1
   else if (q1*dne>r1) {
     if (AC<=r1) {
        LL=Ptilde-AC
        UL=Ptilde+AC
        Flag1=FALSE
      }}
  #Rule 2
   else if (q1*dne<=r1) {
     if(AC<=q1*dne){
       LL=Ptilde-AC
       UL=Ptilde+AC
       Flag1=FALSE
      }}
   }
   Output1=cbind(Phat,Ptilde,AC/dne,LL,UL,runs1)
  return(Output1)
}
```

6. OUTPUT NOTES AND CONCLUSION

The final AJEM code should output estimate(s) for p, including p-hat, p-tilde, upper and lower confidence limits based on p-tilde, the ratio of the confidence interval radius and the distance to the nearest end of p-tilde, and the number of runs completed. If needed, a color code should be included as described in Table 2. Further application of the output requires context of the estimate and the Survivability/Lethality concept under consideration. This context will need to be determined by user-defined standard operating procedures, or by another stakeholder as appropriate.

It is not difficult to think of cases where, for example, the true probability of a mobility loss of function (MLOF) is zero or so close to zero we prefer to call it zero, such as firing a BB gun at a tank. Likewise, it is not difficult to think of cases on the other end of the spectrum, where the probability of kill is at one or extremely close to one, such as a bike hit by a large artillery round. P-hat would be the estimate most desired in those cases, as it can give a value of zero or one. It is possible that the user of these estimates will want to allow for a nonzero, or non-one probability (but close to zero or one). In those cases p-tilde might be preferred. So the user must decide, when considering EPR, how rare an event should be detectable, and whether or not absolute results of zero or one are appropriate.

The other two estimates of use are the bounds of the confidence interval. These can be used as conservative estimates to ensure success in a scenario. For example, suppose the true probability of kill is estimated as $p = 0.2$, and we have an LL = 0.1 and an UL = 0.3 at 95% confidence, all for a single round. To ensure a kill with multiple rounds, the user could choose the LL as the estimate to calculate the number of rounds to get a probability of kill for the scenario of 0.9. Then, assuming one or more rounds are needed to successfully kill, using Pk = 0.2 will give an estimate of $1 - 0.8^n = 0.9$, where n is approximately 11 shots. If Pk = 0.1 then n is approximately 22 shots.

Then the interpretations of the two possible results are as follows:

- Estimate: Using the average estimate of p, we can say the target will be successfully destroyed using an average of 11 shots.
- Confidence Limit: Using the LL estimate of p, we can say that 22 shots will successfully destroy the target with 95% confidence.

Additional estimates could be added to enhance capability. Prediction and tolerance intervals give four more estimates. Their interpretation can be more meaningful with respect to accomplishing a mission or success of an engagement scenario. Using the same example as above (estimates here were not calculated and are only for illustration):

- Prediction Limit: Using the prediction LL estimate (say 25), we can say that we predict on average that the target will be destroyed 90% of the time.
- Tolerance Limit: Using the tolerance LL estimate (say 27), we can say that we predict that the target will be destroyed at least 90% of the time, with 95% confidence.

It is evident that each of these statements increase in strength of certainty that the target will be destroyed. The cost is the increased number of shots needed to meet that level of certainty. The last two statements are a direction some efforts in the Department of Defense are headed. Body armor, for example, uses tolerance limits with respect to test acquisition requirements.

Note that the calculations for LL can be less than zero using the formula, so we have to put a cap on it. Likewise, the UL can be greater than 1 and so needs to be capped at 1. For the design of this algorithm the cost of that cap is comparative to the magnitude of the estimate related to Rule 2, but we could fix that ratio and put the cost in the reduction of the confidence level, which may be less practical.

Consequences of the three choices of estimates are summarized in Table 4. The calculations shown in Table 5 illustrate what we can expect numerically based on the Table 4 perspectives. For all these values of p-true at or near zero, p-tilde, LL, and UL are nearly the same. If we want to use p-hat, then the closer p-true is to zero the more likely we would see exactly zero as the estimate. However, there is a significant possibility that the p-hat is zero when p-true is not zero. Using any of the last three column values would result in pretty much the same analysis whether p-true is 0 or 1/10,000 or 1/1000. If we bump the number of runs up to 10,000, we would likely see better resolution on Event B versus Event C, while Events A and B would be predicted similarly.

Table 4. Consequences of Three Estimate Choices

Estimate	Advantage	Disadvantage
$\hat{p}$, p-hat	Unbiased, not smoothed 0 and 1 are possible	It takes a high sample size to detect a rare or extremely common event probability
$\tilde{p}$, p-tilde	Biased, smoothed Very rare and nearly certain events will still have some nonzero or non-one estimate.	Can be very biased near 0 and 1. Impossible events become possible, certain events become almost certain.
Lower Limit, LL Upper Limit, UL	Biased, Conservative Can use Confidence, Prediction, or Tolerance. 0 or 1 are possible. Provides a push towards more rigorous requirements, designs, etc.	Requirements are more difficult to meet. Can increase costs (computation time, design changes, testing, and other challenges)

Table 5. Zero and Near-Zero Examples

Event	True p	z (2-tail Confidence)	$P(\hat{p} = 0)$	Expected Value of		
				$\tilde{p}$	LL	UL
A	0	1.96 (95%)	100%	0.002	0	0.005
B	0.0001	1.96 (95%)	90%	0.002	0	0.005
C	0.0010	1.96 (95%)	37%	0.003	0	0.006

Picking the cap for Rule 3 will directly affect EPR. If the user needs resolution of 0.001, 1000–5000 runs will likely suffice. If the user needs resolution of 0.00001, then 10,000–50,000 will likely suffice. Confidence level will affect this as well. If the rare event is important to the user, and if computation time is limited, then the number of runs chosen must be planned from this perspective. The user can then adjust the rule parameters in this algorithm to match the required balance between computation and resolution.

7. REFERENCES

Agresti, A. and Coull, B.A. (1998). *Approximate is better than "exact" for interval estimation of binomial proportions*. www.jstor.org/stable/2685469

Appendix A – List of Acronyms

AJEM	Advanced Joint Effectiveness Model
CCDC	U.S. Army Combat Capabilities Development Command
DAC	Data & Analysis Center
EPR	endpoint resolution
LL	lower limit
MLOF	mobility loss of function
UL	upper limit

Appendix B – Distribution List

ORGANIZATION

U.S. Army CCDC Data & Analysis Center
FCDD-DAD-OL/T Resetar-Racine
6896 Mauchly St.
Aberdeen Proving Ground, MD 21005-5068

U.S. Army CCDC Data & Analysis Center
FCDD-DAW-S/C Andres
6509 Wayberry Road
Aberdeen Proving Ground, MD 21005

U.S. Army CCDC Army Research Laboratory
FCDD-RLD-CL/Tech Library
2800 Powder Mill Rd.
Adelphi, MD 20783

Defense Technical Information Center
ATTN: DTIC-O
8725 John J. Kingman Rd.
Fort Belvoir, VA 22060-6218