Al-Qadisiyah Journal of Pure Science

Volume 29 | Number 1 Article 24

2024

Numerical Methods for Statistical Data Generation

Hayder Abdulabbas
Hasan Ministry of Education, hayderabd735@gmail.com

ARTICLE

Numerical Methods for Statistical Data Generation

Hayder A. Hasan

Ministry of Education, Iraq

Abstract

A comprehensive comparison was undertaken between two numerical methods, namely bisection and secant methods, to generate a random variable data for distribution. Two methods were suggested to acquire the initial guesses for secant and interval for the bisection method. The results of 100 cases for the choices of the parameters are shown and a comparison was made according to the number of iterations between the methods was conducted.

Keywords: Monte Carlo simulation, Numerical methods, Root finding, Distribution function

1. Introduction

Statistical inference involves drawing conclusions about a population based on a sample, using probability theory and mathematical formulas. It encompasses hypothesis testing, confidence intervals, and regression analysis to make predictions and decisions in fields like science, economics, and social sciences.

An important tool to study statistical models is generating samples to test statistical distributions. One method for generating these samples is the Monte Carlo simulation. The methodology of generation may sometimes need numerical methods that can provide the results in a timely fashion. In this work we discuss such methods and provide some recommendation to enhance them.

Numerical methods have seen much interest in the academic research. In 1965 Barns [1], provided a method for solving nonlinear system of equations based on the secant method. In 1975, Anderson N [2] gave a property of the secant method which facilitates the computation of extremal eigenvalues of matrices with real eigenvalues. In 1986, Fishman G. [3] gave comprehensive description of a general class of Monte Carlo sampling plans for estimating $g = g(s,T)$, the probability that a specified node s is connected to all nodes in a node set T for a non-directed network $G = (V,E)$ whose arcs are subject to random failure. In 1993, Argyros [4], used the secant method for solving nonlinear operator equations. In 2017, Singh P et al. [5], used the secant method in estimating the frequency of a signal received in a white Gaussian noise environment.

In this work, we use and compare the secant and the bisection methods in generating a sample from a random distribution based on Monte Carlo. We also provide a method to acquire the initial guess and interval for each method.

2. Monte Carlo simulation data sampling [6]

Monte Carlo simulation so often used to assess the goodness of parameter estimators of random variable distributions. To generate the data, the method utilizes random numbers drawn from a standard uniform distribution spanning the interval [0,1]. Each number within this range has an identical likelihood of being generated at any given instance. These generated numbers serve as inputs into the cumulative distribution functions to derive the values of the random variables. The rationale behind this procedure lies in ensuring that the cumulative distribution, denoted as $F(x)$, falls within the range of 0−1, akin to a random number. Consequently, the formula transforms the random number values into variable quantities.

For a random variable X with distribution function $F(x)$, we equate the value p generated form the uniform distribution to $F(x)$ and solve for x. That is we aim to solve

Received 14 March 2024; accepted 11 April 2024.
Available online 28 September 2024
E-mail address: hayderabd735@gmail.com.

https://doi.org/10.29350/2411-3514.1262

$$F(x)-p=0$$

The above equation is not always easy to solve algebraically, and therefore numerical methods are implemented to acquire the solution. The larger the size of data, the more time it takes for those methods to generate the data. It is therefore vital to use more efficient numerical methods to save time.

An example of such distribution function is that of the Double Weighted Length Biased Exponential Distribution [7].

The pdf is given by

$$f(x,\theta,c)=\frac{(c+1)^6}{24\left((c+1)^6-5c-(c+1)\right)}x^4\theta^5e^{-\theta x}\left[1-(\theta cx+1)e^{-\theta cx}\right]$$

With

$$F(x)=\frac{(c+1)^6}{24\left((c+1)^6-5c-(c+1)\right)}\left[-\theta^4x^4e^{-\theta x}-4\theta^3x^3e^{-\theta x}-12\theta^2x^2e^{-\theta x}-24\theta xe^{-\theta x}-24e^{-\theta x}+\frac{c}{c+1}\theta^5x^5e^{-\theta x(c+1)}+\frac{6c+1}{(c+1)^2}\theta^4x^4e^{-\theta x(c+1)}+\frac{24c+4}{(c+1)^3}\theta^3x^3e^{-\theta x(c+1)}+\frac{72c+12}{(c+1)^4}\theta^2x^2e^{-\theta x(c+1)}+\frac{144c+24}{(c+1)^5}\theta xe^{-\theta x(c+1)}+\frac{144c+24}{(c+1)^6}e^{-\theta x(c+1)}+\frac{24(c+1)^6-144c-24}{(c+1)^6}\right]$$

Th distribution has two parameters θ and c and $x>0$.

Fig. 1 shows the behavior of F and the values of the sample data as a function of $F-p$

3. Numerical methods

Two numerical methods are considered in this work, namely Bisection and secant methods.

3.1. Bisection method [8]

The bisection method is a simple numerical technique used to find the root of a continuous function within a given interval. It is based on the intermediate value theorem, which states that if a continuous function has different signs at two points within an interval, then it must have at least one root within that interval.

Here's how the bisection method works:

1. Initial Setup: You start with an interval $[a,b]$ where the function changes sign (i.e., $f(a)$ and $f(b)$ have opposite signs), indicating a root might exist within this interval.
2. Bisection: Calculate the midpoint $c=\frac{(a+b)}{2}$ of the interval $[a,\ b]$. Evaluate the function at this midpoint, $f(c)$.
3. Update Interval: Check the sign of $f(c)$. If $f(c)$ has the same sign as $f(a)$, replace a with c; otherwise, replace b with c. In this way, you narrow down the interval containing the root.
4. Iterate: Repeat steps 2 and 3 until the interval becomes sufficiently small, or until the function value at the midpoint is sufficiently close to zero.
5. Result: The final interval $[a,b]$ will contain the root. Typically, the midpoint of this interval is taken as an approximation to the root.

The bisection method guarantees convergence to a root if the function is continuous on the interval $[a,\ b]$ and if $f(a)$ and $f(b)$ have opposite signs, ensuring the existence of a root within the interval. However, it converges relatively slowly compared to some

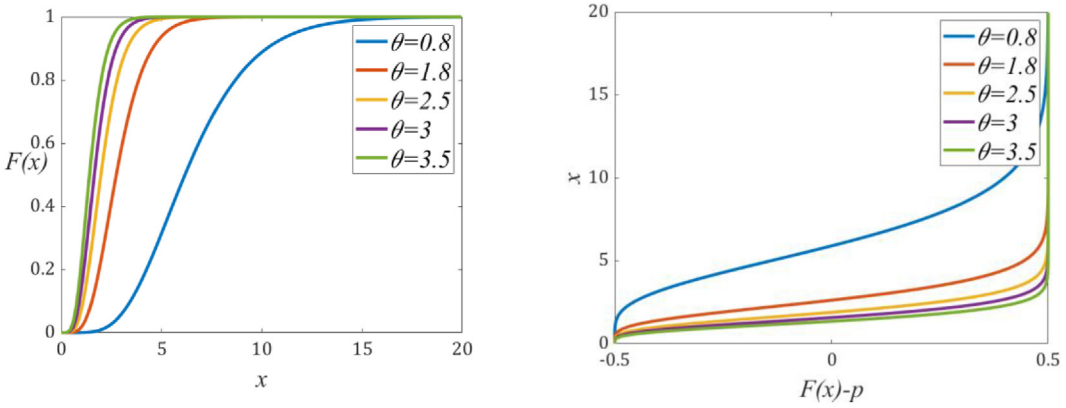

Fig. 1. The cumulative distribution function (on the left), and the values of the sample data as a function of $F-p$. Here $p=0.5$.

other root-finding methods, particularly when the function has rapid changes or multiple roots within the interval. Nonetheless, it is a reliable and straightforward method for finding roots of continuous functions.

3.2. Secant method [8]

The secant method is another numerical technique used for finding the root of a continuous function. It is similar to the bisection method but instead of narrowing down intervals, it uses successive secant lines to approximate the root.

Here's how the secant method works:

1. Initial Guesses: Start with two initial guesses, x_0 and x_1, which may or may not bracket the root (i.e., $f(x_0)$ and $f(x_1)$ have opposite signs).
2. Secant Line: Construct a secant line passing through the points $(x_0, f(x_0))$ and $(x_1, f(x_1))$.
3. Intersection: Find the x-coordinate of the intersection of the secant line with the x-axis. This point serves as the next approximation to the root.
4. Update Guesses: Update x_0 and x_1 such that x_1 becomes the previous approximation, and the new approximation becomes the intersection point.
5. Iteration: Repeat steps 2–4 until the desired level of accuracy is achieved or until convergence criteria are met.

The secant method typically converges faster than the bisection method but may not always converge, especially if the initial guesses are not chosen appropriately or if the function behaves irregularly. However, it is computationally simpler than some other methods like Newton's method since it doesn't require calculating derivatives. Despite its limitations, the secant method is widely used for its simplicity and effectiveness in finding roots of functions.

4. Root finding methodology and algorithm

Since the support of x is all non-negative numbers, a method to acquire the interval and initial guesses for the bisection and secant methods is needed.

The initial interval for the bisection method is determined by starting with a lower bound $x_0 = 0$ and incrementing number x_1 by 1 until the sign of the function changes (i.e., the function evaluates to different signs at x_0 and x_1). This process ensures that the initial interval $[x_0, x_1]$ contains a root, as the function changes sign within this interval.

For the secant method a similar algorithm to that of the bisection an initial guess $x_0 = 0$ and incrementing another guess x_1 until the sign of the function changes. The interval $[x_0, x_1]$ with the changed sign of the function at the endpoints serves as the initial approximation for the root.

In both methods, the choice of the initial interval is crucial as it provides a starting range for the algorithm to search for the root. By selecting an interval where the function changes sign, we ensure that the root lies within this interval, increasing the likelihood of convergence to the root. The methods are used to find the roots of $F(x) - p$ and the results are given in Table 1.

Another modification for both methods is to put the lower bound for the bisection to be $x_1 - 1$ which make the interval $[x_1 - 1, x_1]$ which also contain the root. And also use the $x_1 - 1$ and x_1 as the first and second guesses for the secant method. The results of this implementation are given in Table 2

MATLAB is used to acquire the interval and initial interval and initial guesses and the code is given the Code 2 in the appendix for a function F.

The CDF function is generated using Code 1 in the appendix under five choices of $\theta = 0.8, 1.8, 2.5, 3, 3.5$ and $c = 1, 2$. The function $F - p$ is then generated using Five choices of $p = 0.1, 0.3, 0.5, 0.7, 0.9$. The root to a precision of 10^{-6} and number of iterations required to acquire it for each method is then found using Code 2 in the appendix.

An sum of 100 cases are solved and Table 1 shows the results for when the interval bounds and guesses are 0 and x_1 and Table 2 shows the results for when the interval bounds and guesses are $x_1 - 1$ and x_1

5. Result discussion

A through comparison was conducted between two methods for acquire the root of the function $F(x) - p$ where F is a cumulative distribution function and $p \in (0, 1)$. Implementing the interval $[0, x_1]$, where x_1 is the increment of 0 by 1 until the sign of $F - p$ changes, to be the initial guess for both methods the results show a significant advantage for the secant method over bisection as shown in Table 1 where the iteration for the bisection method ranged up to 23 iterations and 8 for the secant method. An improvement is noticed when implementing the interval $[x_1 - 1, x_1]$ where the iterations of the bisection iterations were reduced to 18. In the case of the secant method, a slight improvement is seen as shown in Table 2.

Table 1. The performance of both bisection and secant methods for the first initial guess method.

θ	c	p	Bisection		Secant	
			Root	Iterations	Root	Iterations
0.8	1	0.1	3.451711	21	3.451711	6
1.8	1	0.1	1.534093	20	1.534094	7
2.5	1	0.1	1.534093	20	1.534094	7
3	1	0.1	0.920457	19	0.920456	5
3.5	1	0.1	0.788962	19	0.788962	7
0.8	1	0.3	4.91059	22	4.910591	4
1.8	1	0.3	2.182485	21	2.182485	5
2.5	1	0.3	2.182485	21	2.182485	5
3	1	0.3	1.309491	20	1.309491	5
3.5	1	0.3	1.12242	20	1.122421	4
0.8	1	0.5	6.16895	22	6.16895	5
1.8	1	0.5	2.741756	21	2.741756	4
2.5	1	0.5	2.741756	21	2.741756	4
3	1	0.5	1.645053	20	1.645053	5
3.5	1	0.5	1.410047	20	1.410046	5
0.8	1	0.7	7.655545	22	7.655546	4
1.8	1	0.7	3.402465	21	3.402465	4
2.5	1	0.7	3.402465	21	3.402465	4
3	1	0.7	2.041479	21	2.041479	6
3.5	1	0.7	1.749839	20	1.749839	4
0.8	1	0.9	10.24002	23	10.24002	5
1.8	1	0.9	4.551122	22	4.551122	5
2.5	1	0.9	4.551122	22	4.551122	5
3	1	0.9	2.730673	21	2.730673	5
3.5	1	0.9	2.340576	21	2.340577	8
0.8	2	0.1	3.157596	21	3.157595	7
1.8	2	0.1	1.403375	20	1.403376	7
2.5	2	0.1	1.403375	20	1.403376	7
3	2	0.1	0.842025	19	0.842025	6
3.5	2	0.1	0.721736	19	0.721736	7
0.8	2	0.3	4.619529	22	4.619528	4
1.8	2	0.3	2.053124	21	2.053124	5
2.5	2	0.3	2.053124	21	2.053124	5
3	2	0.3	1.231874	20	1.231874	5
3.5	2	0.3	1.055892	20	1.055892	5
0.8	2	0.5	5.898354	22	5.898354	4
1.8	2	0.5	2.62149	21	2.621491	4
2.5	2	0.5	2.62149	21	2.621491	4
3	2	0.5	1.572894	20	1.572894	5
3.5	2	0.5	1.348195	20	1.348195	5
0.8	2	0.7	7.411403	22	7.411402	4
1.8	2	0.7	3.293956	21	3.293956	4
2.5	2	0.7	3.293956	21	3.293956	4
3	2	0.7	1.976375	20	1.976374	3
3.5	2	0.7	1.694036	20	1.694035	4
0.8	2	0.9	10.0311	23	10.0311	5
1.8	2	0.9	4.458267	22	4.458267	6
2.5	2	0.9	4.458267	22	4.458267	6
3	2	0.9	2.67496	21	2.67496	6
3.5	2	0.9	2.292823	21	2.292823	8

Table 2. The performance of both bisection and secant methods for the second initial guess method.

θ	c	p	Bisection		Secant	
			Root	Iterations	Root	Iterations
0.8	1	0.1	3.451711	19	3.451711	4
1.8	1	0.1	1.534093	19	1.534094	5
2.5	1	0.1	1.534093	19	1.534094	5
3	1	0.1	0.920457	19	0.920456	5
3.5	1	0.1	0.788962	19	0.788962	7
0.8	1	0.3	4.91059	19	4.910591	3
1.8	1	0.3	2.182485	19	2.182485	3
2.5	1	0.3	2.182485	19	2.182485	3
3	1	0.3	1.309491	19	1.309491	3
3.5	1	0.3	1.12242	19	1.122421	4
0.8	1	0.5	6.16895	19	6.16895	3
1.8	1	0.5	2.741755	19	2.741756	4
2.5	1	0.5	2.741755	19	2.741756	4
3	1	0.5	1.645053	19	1.645053	4
3.5	1	0.5	1.410047	19	1.410046	5
0.8	1	0.7	7.655545	19	7.655546	4
1.8	1	0.7	3.402465	19	3.402465	5
2.5	1	0.7	3.402465	19	3.402465	5
3	1	0.7	2.041478	19	2.041479	5
3.5	1	0.7	1.749839	19	1.749839	5
0.8	1	0.9	10.24002	19	10.24002	4
1.8	1	0.9	4.551122	19	4.551122	5
2.5	1	0.9	4.551122	19	4.551122	5
3	1	0.9	2.730674	19	2.730673	5
3.5	1	0.9	2.340577	19	2.340577	6
0.8	2	0.1	3.157596	19	3.157595	4
1.8	2	0.1	1.403375	19	1.403376	5
2.5	2	0.1	1.403375	19	1.403376	5
3	2	0.1	0.842025	19	0.842025	6
3.5	2	0.1	0.721736	19	0.721736	7
0.8	2	0.3	4.619529	19	4.619528	3
1.8	2	0.3	2.053124	19	2.053124	3
2.5	2	0.3	2.053124	19	2.053124	3
3	2	0.3	1.231874	19	1.231874	3
3.5	2	0.3	1.055892	19	1.055892	4
0.8	2	0.5	5.898355	19	5.898354	3
1.8	2	0.5	2.621491	19	2.621491	4
2.5	2	0.5	2.621491	19	2.621491	4
3	2	0.5	1.572894	19	1.572894	4
3.5	2	0.5	1.348195	19	1.348195	5
0.8	2	0.7	7.411403	19	7.411402	4
1.8	2	0.7	3.293956	19	3.293956	5
2.5	2	0.7	3.293956	19	3.293956	5
3	2	0.7	1.976375	19	1.976374	3
3.5	2	0.7	1.694036	19	1.694035	5
0.8	2	0.9	10.0311	19	10.0311	4
1.8	2	0.9	4.458266	19	4.458267	5
2.5	2	0.9	4.458266	19	4.458267	5
3	2	0.9	2.67496	19	2.67496	5
3.5	2	0.9	2.292823	19	2.292823	6

Funding

Self-funding.

Appendix A. Appendix of Codes

1. Plotter and CDF Generator

```
clc;clear;
fontsize=18;
c=2;
thetaa=[0.8;1.8;2.5;3;3.5];
n=10;                %sample size
R=500;                %simulation size
c1=c+1;c2=(6*c)+1;
p=0;
%% Dummy vectors

%% Equation
for i=1:5
    theta=thetaa(i)
syms y
Mul=((c1^6)/(24*(c1^6-6*c-1)));
M01=-theta^4*y^4*exp(-theta*y);
M02=-4*theta^3*y^3*exp(-theta*y);
M03=-12*theta^2*y^2*exp(-theta*y);
M04=-24*theta*y*exp(-theta*y);
M05=-24*exp(-theta*y);
M06=(c/c1)*theta^5*y^5*exp(-theta*y*c1);
M07=(c2/(c1^2))*theta^4*y^4*exp(-theta*y*c1);
M08=(4*c2/(c1^3))*theta^3*y^3*exp(-theta*y*c1);
M09=(12*c2/(c1^4))*theta^2*y^2*exp(-theta*y*c1);
M10=(24*c2/(c1^5))*theta*y*exp(-theta*y*c1);
M11=(24*c2/(c1^6))*exp(-theta*y*c1);
M12=1/Mul;
M=Mul*(M01+M02+M03+M04+M05+M06+M07+M08+M09+M10+M11+M12)-p;
F=matlabFunction(M);
func=matlabFunction(diff(M));
a=0:0.15:20;
b=F(a);
s=func(a);
plot(a,b,'LineWidth',3)
hold on
end
legend({...
    "\theta="+thetaa(1),...
    "\theta="+thetaa(2),...
    "\theta="+thetaa(3),...
    "\theta="+thetaa(4),...
    "\theta="+thetaa(5)},...
    'FontSize',24,'location','northeast','FontName','Times      New
Roman',...
    'FontAngle','italic')
set(gca,'FontSize',fontsize,'FontName','Cambria math')
xlabel('x',       'FontSize',       24,'FontName','Times       New
Roman','FontAngle','italic')
ylabel('F(x)',  'FontSize',  24,  'Rotation',0,'FontName','Times  New
Roman','FontAngle','italic')
```

2. Interval Choice and solver

```
  f = F;
  x0 = 0;  % Initial guess
  x1 = 1;  % Initial increment
  while sign(f(x0)) == sign(f(x1))
      x1 = x1 + 1;  % Increment until sign changes
  end
  %x0= x1 - 1
  initial_guesses = [x0, x1];
  initial_guess_fixed_point = x0;
  tol = 1e-9;
  max_iter = 100;
  [root_bisection,   iter_bisection]   =   findRoot('bisection',   f,
initial_guesses, tol, max_iter);
  [root_secant,     iter_secant]     =     findRoot('secant',     f,
initial_guesses, tol, max_iter);
  disp('Bisection method:');
  disp(['Root: ', num2str(root_bisection)]);
  disp(['Iterations: ', num2str(iter_bisection)]);
  disp(' ');
  disp('Secant method:');
  disp(['Root: ', num2str(root_secant)]);
  disp(['Iterations: ', num2str(iter_secant)]);
  disp(' ');
  Final=[root_bisection iter_bisection root_secant iter_secant];
  function   [root,   num_iterations]   =   findRoot(method,   func,
initial_guesses, tol, max_iter)
      switch lower(method)
          case 'bisection'
              [root,  num_iterations,  converged]  =  bisection(func,
initial_guesses(1), initial_guesses(2), tol, max_iter);
          case 'secant'
              [root,   num_iterations,   converged]   =   secant(func,
initial_guesses(1), initial_guesses(2), tol, max_iter);
                  otherwise
              error('Invalid    method.    Please    choose    either
''bisection'', ''secant'', or ''fixed_point''.');
      end

      if ~converged
          warning(['Method ', method, ' did not converge within the
maximum number of iterations.']);
      end
  end
  function [root, num_iterations, converged] = bisection(func, a, b,
tol, max_iter)
      iter = 0;
      while iter < max_iter
          c = (a + b) / 2;
          if func(c) == 0 || (b - a) / 2 < tol
              root = c;
              num_iterations = iter;
              converged = true;
              return;
          end
          iter = iter + 1;
          if sign(func(c)) == sign(func(a))
              a = c;
          else
              b = c;
          end
      end
      root = NaN;
      num_iterations = max_iter;
      converged = false;
  end
  function [root, num_iterations, converged] = secant(func, x0, x1,
tol, max_iter)
      iter = 0;
      while iter < max_iter
          f0 = func(x0);
          f1 = func(x1);
          x_next = x1 - f1 * (x1 - x0) / (f1 - f0);
          if abs(x_next - x1) < tol
              root = x_next;
              num_iterations = iter;
              converged = true;
              return;
          end
          x0 = x1;
          x1 = x_next;
          iter = iter + 1;
      end
      root = NaN;
      num_iterations = max_iter;
      converged = false;
  end
```

References

[1] Barnes JGP. An algorithm for solving non-linear equations based on the secant method. Comput J Apr. 1965;8(1):66–72. https://doi.org/10.1093/comjnl/8.1.66.
[2] Anderson N. On computing eigenvalues of matrices with real eigenvalues by the secant method. CM-P00069400; 1975.
[3] Fishman GS. A Monte Carlo sampling plan for estimating network reliability. Oper Res Aug. 1986;34(4):581–94. https://doi.org/10.1287/opre.34.4.581.
[4] Argyros IK. On the secant method. Publ Math Debrecen 1993; 43(3–4):223–38.
[5] Singh P, Singhal A, Joshi SD. An efficient ML frequency estimation of a sinusoid using the Secant method. In: 2017 IEEE international conference on advanced networks and telecommunications systems (ANTS); Dec. 2017. p. 1–5. https://doi.org/10.1109/ANTS.2017.8384150.
[6] Kroese DP, Taimre T, Botev ZI. Handbook of Monte Carlo methods. John Wiley & Sons; 2013.
[7] Elaiwi Ubaid Intisar. Double weighted functions of some probability distributions. Baghdad: MsC, Mustansiriyah University; 2023.
[8] Burden RL, Faires JD. Numerical analysis. eighth ed. Thomson Brooks/Cole; 2005.