Analyzing Equilibrium Points in Population Models Using Newton's Method and Fixed-Point Iteration

Colton Blackwell

**Assume $G(N) = r(1-N)-N$; $G'(N) = r-2rN-1$

Purpose

This report explores and compares different numerical methods, specifically Newton's method and fixed-point iteration, to analyze equilibrium points in discrete dynamical systems. Focusing on a population growth model described by the equation $N_{t+1} = rN_t(1-N_t)$, the study identifies equilibrium points and determines their realistic ranges for various values of r. Newton's method is employed to find the largest realistic root for $r \in [0.1,4]$, with the functions G(N) and G′(N) defined as $rN(1-N)-N$ and $r-2rN-1$, respectively. The numerical results are then graphically compared with the analytical solution. Additionally, the fixed-point method is applied for selected r values ($r=0.5, r=1.5, r=2$, and some $r \in [3,3.4]$), observing different convergence and divergence patterns. These patterns are illustrated graphically and explained in terms of population dynamics and the concept of chaos in discrete dynamical systems.

By analyzing equilibrium points and stability in population models, it aids in predicting and managing wildlife populations and disease spread. The numerical methods studied, Newton's method and fixed-point iteration, have broad applications in various fields. Understanding convergence, divergence, and chaos helps design robust models and strategies to handle real-world unpredictability. Additionally, the experiment highlights how rounding errors in finite-precision arithmetic can cause algorithm instability, underscoring the need to design better algorithms to mitigate such issues.

Observations

To find the equilibrium points for the given population growth equation, the equation $N^*=F(N^*)$ needed to be set up and solved. First, the equation $N^*=rN^*(1-N^*)$ was rearranged and simplified to $N^*(rN^*-r+1)=0$. This equation gave two solutions: $N^*=0$ and $rN^*-r+1=0$. Solving the second equation for N* resulted in $N^*=(r-1)/r$. This meant that the equilibrium points were $N^*=0$ and $N^*=(r-1)/r$. For these equilibrium points to have realistic meaning in the context of population growth, $N^*=0$ represented the extinction of the species and was a realistic equilibrium point for any value of r. Additionally, $N^*=(r-1)/r$ was required to be between 0 and 1 since population sizes are typically normalized to fall within this range. For the lower bound, $(r-1)/r \geq 0$ implied $r \geq 1$; therefore, $N^*=(r-1)/r$ was a realistic equilibrium point for $r \geq 1$.

To find the largest root for $r \in [0.1,4]$, the functions G and G' (**See function definitions above) were prepared for use with Newton's method. The initial guess was set to 0.5, this being the midpoint of the normalized population size range [0, 1]. Observing the results shown in Fig. 1, Newton's method successfully found the correct root, as it matches the analytical solution $N^*=(r-1)/r$.

Using the fixed point method while observing the behavior of the function $g(x)=rx(1-x)$ for different values of r reveals interesting dynamics. When r = 0.5, the function converges quickly to a stable fixed point at x = 0, benefiting from the condition r < 1 that ensures stability. Increasing r to 1.5 still results in convergence, albeit with the fixed point shifting to a non-zero value; however, this convergence may be slower due to the derivative g'(x) being closer to 1. At r=2, the function converges to a fixed point between 0 and 1, showcasing varying rates of convergence while maintaining stability. As r enters the range [3,3.4], the function's behavior becomes more complex, potentially exhibiting periodic behavior or chaos. For instance, at r=3.2, periodic oscillations can occur, highlighting the non-linear dynamics and unpredictability that can arise in this range of r values.

Understanding

To understand the equivalence shown in Fig. 1 and the relationship between the two functions, it is important to note that Newton's method seeks the roots of the equation G(N)=0, which correspond to the equilibrium points of the system. The function G(N) used in Newton's method for solving the equation $rN_t(1-N_t)-N_t=0$ simplifies to $G(N)=rN-rN^2-N$. Comparing this with the analytical solution $N_t=(r-1)/r$, we substitute N_t with N to get $N=(r-1)/r$. Substituting this into G(N), we find that $G((r-1)/r)$ simplifies to zero, confirming that G(N) in Newton's method is equivalent to the analytical solution at the equilibrium points of the population growth system.

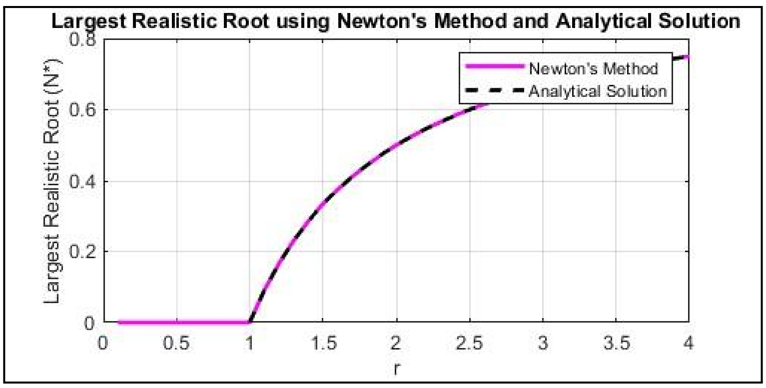

Fig. 1

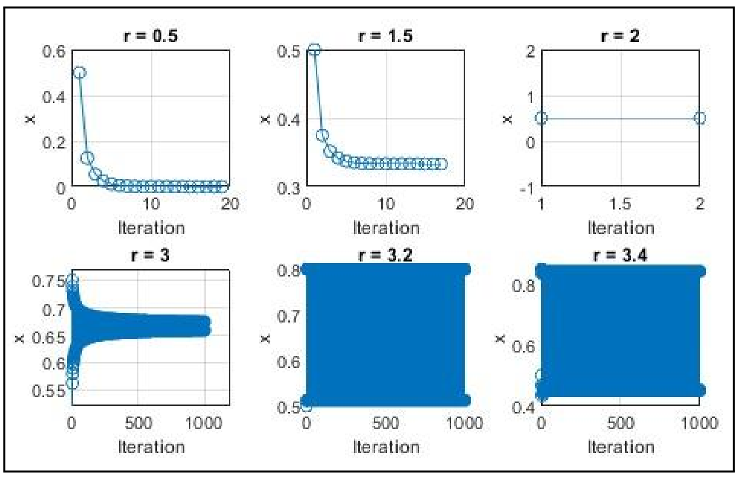

Fig. 2

Editor - C:\Users\colto\Downloads\fixedpt_test.m

newton.m | newton_population_growth.m | fixedpt.m | fixedpt_test.m | +

```
% Define the function for fixed point iteration
fixedpt_function = @(r, x) r * x * (1 - x);

% Parameters
xguess = 0.5;
tol = 1e-6;
maxiter = 1000;

% Test different values of r
r_values = [0.5, 1.5, 2, 3, 3.2, 3.4];

% Create a figure to hold all the plots
figure;
hold on;

% Iterate over each r value
for i = 1:length(r_values)
    r = r_values(i);
    gfunc = @(x) fixedpt_function(r, x);
    [xfinal, niter, xlist] = fixedpt(gfunc, xguess, tol, maxiter);

    subplot(2, 3, i);
    plot(1:niter+1, xlist, '-o');
    title(['r = ', num2str(r)]);
    xlabel('Iteration');
    ylabel('x');
    grid on;
end

hold off;
```

Editor - C:\Users\colto\Downloads\newton_population_growth.m

newton.m | newton_population_growth.m | fixedpt.m | fixedpt_test.m

```
function newton_population_growth()
    % Define the range of r
    r_values = 0.1:0.1:4;

    % Store the largest realistic root for each r
    largest_root = zeros(size(r_values));

    % Iterate over each r
    for idx = 1:length(r_values)
        r = r_values(idx);

        % Define G(N) and G'(N) for current r
        G = @(N) r * N * (1 - N) - N;
        G_prime = @(N) r - 2 * r * N - 1;

        % Initial guess
        N0 = 0.5;

        % Ensure the initial guess avoids problematic points
        [root, ~, ~] = safe_newton(G, G_prime, N0);

        % Update largest root if the current root is larger
        largest_root(idx) = max(largest_root(idx), root);
    end

    % Plot the results
    figure;
    plot(r_values, largest_root, 'm-', 'LineWidth', 2);
    hold on;
    plot(r_values(r_values >= 1), (r_values(r_values >= 1) - 1) ./ r_values(r_values >= 1), 'k--', 'LineWidth', 2);
    xlabel('r');
    ylabel('Largest Realistic Root (N*)');
    legend('Newton''s Method', 'Analytical Solution');
    title('Largest Realistic Root using Newton''s Method and Analytical Solution');
    grid on;
end

function [root, iter, xlist] = safe_newton(func, pfunc, xguess, tol)
    % Safe version of Newton's method to handle zero derivatives and adjust initial guesses
    if nargin < 4
        tol = 1e-6;
    end

    % Initialize variables
    x = xguess;
    iter = 0;
    max_iter = 100;
    xlist = x;

    while iter < max_iter
        fx = feval(func, x);
        fpx = feval(pfunc, x);

        % Check for zero derivative an
        if abs(fpx) < tol
            x = x + rand() * 0.1; % Ad
            iter = iter + 1;
            continue;
        end

        x_new = x - fx / fpx;

        if abs(x_new - x) < tol
            break;
        end

        x = x_new;
        xlist = [xlist; x];
        iter = iter + 1;
    end

    root = x;
 end
```

Editor - C:\Users\colto\Downloads\fixedpt.m

newton.m | newton_population_growth.m | fixedpt.m | fixedpt_test.m | +

```
function [xfinal, niter, xlist] = fixedpt(gfunc, xguess, tol, maxiter)
% FIXEDPT: Fixed point iteration for x=gfunc(x).
%
%  Sample usage:
%     [xfinal, niter, xlist] = fixedpt(gfunc, xguess, tol, maxiter)
%
%  Input:
%     gfunc   - fixed point function
%     xguess  - initial guess at the fixed point
%     tol     - convergence tolerance (OPTIONAL, defaults to 1e-6)
%     maxiter - maximum number of iterations (OPTIONAL, defaults to 1000)
%
%  Output:
%     xfinal  - final estimate of the fixed point
%     niter   - number of iterations to convergence
%     xlist   - list of iterates, an array of length 'niter'

% First, do some error checking on parameters.
if nargin < 2
  fprintf(1, 'FIXEDPT: must be called with at least two arguments');
  error('Usage: [xfinal, niter, xlist] = fixedpt(gfunc, xguess, [tol], [maxiter])');
end
if nargin < 3, tol = 1e-6; end
if nargin < 4, maxiter = 1000; end

% fcnchk(...) allows a string function to be sent as a parameter, and
% converts it to the correct type to allow evaluation by feval().
gfunc = fcnchk(gfunc);
x = xguess;
xlist = x;

niter = 0;
done  = 0;
while ~done
  xnew  = feval(gfunc, x);
  xlist = [xlist; xnew];  % create a list of x-values
  niter = niter + 1;
  if abs(x - xnew) < tol || niter >= maxiter  % stopping tolerance for x or max iterations
    done = 1;
  end
  x = xnew;
end
xfinal = xnew;
end
```