```
% MATH 121, Spring 2023
% Author: Mario Bencomo, CSU Fresno State
% Template code for Report 1A
% Setting plotting specifications
set(0, 'defaultaxesfontsize',18,'defaultaxeslinewidth',1.2,...
       'defaultlinelinewidth',1.0,'defaultpatchlinewidth',1.0,...
       'defaulttextfontsize',18);
MAX_ITER = 30;

%% Problem 1

true_error = zeros(1, MAX_ITER);

ETOL = 1e-15;
FTOL = 1e-15;
%% Demo: Iterative approach to division, i.e., computing 1/z with z>0
z = 6;
f = @(x) x.^2 - z;
df = @(x) x*2;
a = 0;
b = z;
VERBOSE = true;
[r_B,err_B] = my_bisect(a,b,f,MAX_ITER,ETOL,FTOL,VERBOSE);
[r_N,err_N] = my_newton(z,f,df,MAX_ITER,ETOL,FTOL,VERBOSE);
[r_P,err_P] = my_secant(0,z,f,MAX_ITER,ETOL,FTOL,VERBOSE);
figure;
semilogy(abs(r_B-sqrt(6)),'-ob'); hold on;
semilogy(err_B,'--b');
semilogy(abs(r_N-sqrt(6)),'-ok'); hold on;
semilogy(err_N,'--b');
semilogy(abs(r_P-sqrt(6)),'-og');
semilogy(err_P,'--b');


title('Error analysis of methods');
xlabel('iteration count');
legend({'true err','est err'},'Interpreter','latex');


%% Problem 2


%% Implementation of methods
function [r,err] = my_bisect(a0,b0,f,MAX_ITER,ETOL,FTOL,VERBOSE)
% function my_bisect(a0,b0,f,MAX_ITER,ETOL,FTOL,VERBOSE)
%
% This is an implementation of the bisection method for solving f(x)=0.
% Inputs:
%         a0 = left side of search interval
%         b0 = right side of search interval
%          f = function for which we want to find the root of
%   MAX_ITER = maximum number of iterations for stopping criterion
%       ETOL = tolerance on error bound
%       FTOL = tolerance on abs(f(r))
%    VERBOSE = if true will output and plot intermediate results
% Outputs:
%     r = approximation of root; if VERBOSE=true, then it will contain
%         intermediate approximations too
%   err = b-r, error estimate of the root approximation; if VERBOSE=true, then
%         it will contain intermediate errors
if a0 >= b0
    disp('Error(my_bisection): a0<b0 is not true! Exiting!')
    return
end
a = a0;
b = b0;
fa = f(a);
fb = f(b);
if sign(fa)*sign(fb) > 0
    disp('Error(my_bisection): f(a0) and f(b0) are of the same sign! Exiting!')
    return
end
r = (a+b)/2;
count = 0;
err = b-r;
fr = f(r);
r_hist = r;
err_hist = err;
if VERBOSE
    fprintf('\nFrom my_bisection')
    fprintf('\ncount root    error est  f(r)\n');
    fprintf('%5d %6.5f %6.4e %6.4e\n',[count,r,err,fr]);
    %pause
end
while err>ETOL && count<MAX_ITER && abs(fr)>FTOL
    if sign(fb)*sign(fr)<=0
        a = r;
        fa = fr;
    else
        b = r;
        fb = fr;
    end
    count = count+1;
    r = (a+b)/2;
    err = b-r;
    fr = f(r);
    if VERBOSE
        fprintf('%5d %6.5f %6.4e %6.4e\n',[count,r,err,fr]);
        r_hist = [r_hist r];
        err_hist = [err_hist err];
        %pause
    end
end
if count==MAX_ITER
    disp('Warning(my_bisection): terminated after max number of iterations.')
end
if VERBOSE
    r = r_hist;
    err = err_hist;
end
end


function [r,err] = my_newton(x0,f,df,MAX_ITER,ETOL,FTOL,VERBOSE)
% function my_Newton(x0,f,df,MAX_ITER,TOL,VERBOSE)
%
% This is an implementation of Newton's method for solving f(x)=0.
% Inputs:
```

```
%         x0 = initial approximation
%          f = anonymous function for which we want to find the root of
%         df = anonymous function which computes derivative of f
%   MAX_ITER = maximum number of iterations for stopping criterion
%       ETOL = tolerance on error bound
%       FTOL = tolerance on abs(f(r))
%    VERBOSE = if true will output and plot intermediate results
% Outputs:
%     r = approximation of root; if VERBOSE=true, then it will contain
%         intermediate approximations too
%   err = error estimate of the root approximation; if VERBOSE=true, then
%         it will contain intermediate errors
r = x0;
count = 0;
err = abs(f(r));
r_hist = [];
err_hist = [];

if VERBOSE
    fprintf('\nFrom my_Newton')
    fprintf('\ncount root    error est  f(r)\n');
    fprintf('%5d %6.5f %6.4e %6.4e\n',[count,r,err,f(r)]);
%     pause
end

while (err > ETOL) && (count < MAX_ITER) && (abs(f(r)) > FTOL)
    r = r - f(r)/df(r);
    err = abs(f(r));
    count = count + 1;



    if VERBOSE
        fprintf('%5d %6.5f %6.4e %6.4e\n',[count,r,err,f(r)]);
        r_hist = [r_hist r];
        err_hist = [err_hist err];
%         pause
    end
end

if count == MAX_ITER
    disp('Warning(my_Newton): terminated after max number of iterations.')
end

if VERBOSE
    r = r_hist;
    err = err_hist;
end
end



function [r,err] = my_secant(x0,x1,f,MAX_ITER,ETOL,FTOL,VERBOSE)
% function my_secant(x0,x1,f,MAX_ITER,ETOL,FTOL,VERBOSE)
%
% This is an implementation of the secant method for solving f(x)=0.
% Inputs:
%      x0,x1 = initial approximations
%          f = anonymous function for which we want to find the root of
%   MAX_ITER = maximum number of iterations for stopping criterion
%       ETOL = tolerance on error bound
%       FTOL = tolerance on abs(f(r))
%    VERBOSE = if true will output and plot intermediate results
% Outputs:
%     r = approximation of root; if VERBOSE=true, then it will contain
%         intermediate approximations too
%   err = error estimate of the root approximation; if VERBOSE=true, then
%         it will contain intermediate errors
count = 0;
err = inf;
r_hist = [];
err_hist = [];
if nargin < 7
    VERBOSE = false;
end

f(x1)

if VERBOSE
    fprintf('\nFrom my_secant')
    fprintf('\ncount root    error est  f(r)\n');
    fprintf('%5d %6.5f %6.4e %6.4e\n',[count,x1,err,f(x1)]);
end

% initialize variables
r_old = x0;
r = x1;
f_old = f(r_old);
f_curr = f(r);

% main loop
while count < MAX_ITER && err > ETOL && abs(f_curr) > FTOL
    count = count + 1;

    % update variables
    delta_r = -f_curr * (r - r_old) / (f_curr - f_old);
    r_old = r;
    r = r + delta_r;

    % update error estimate
    err = abs(delta_r);

    % update function values
    f_old = f_curr;
    f_curr = f(r);

    % verbose output
    if VERBOSE
        fprintf('%5d %6.5f %6.4e %6.4e\n',[count,r,err,f_curr]);
        r_hist = [r_hist r];
        err_hist = [err_hist err];
    end
end

if count == MAX_ITER
    disp('Warning(my_secant): terminated after max number of iterations.')
end
```

```
if VERBOSE
    r = r_hist;
    err = err_hist;
end
end
```