Week 1: Introduction – A lower bounded random walk
Solutions

A Simulation of a realization of the process. The function is already provided on the wiki-page of the course: `http://www.seas.upenn.edu/~ese303/block_0_introduction/code/ssas_gambling.m`. Recall that functions must be saved in an m-file with the same name as the function.

```
% Function encapsulating the loop to simulate bets.
%
% At each time instant t, player bets $b and wins with probability p,
% receiving a payoff of $b. Otherwise, the player loses the bet. The player
% starts with wealth w(0) and
% plays until his wealth is depleted. To avoid infinite loops we also add
% a restriction max_t times on the number of bets placed.
%
% The function returns a vector w with the history of the bets and a scalar
% t with the number of bets placed. If the player didn't go bankrupt,
% t = max_t.
function [w, h] = gambling(w_0, b, p, max_t)

t = 1;  % Time index
h = 0;  % Boolean to indicate bankruptcy

w = zeros(max_t+1,1);
w(1) = w_0;

while ( (w(t) > 0) && (t <= max_t) )    % Halt the iteration when wealth
                                        % is depleted or after max_t bets
   x = random('bino',1,p);              % Draw random number determining bet outcome
                                        % x = 1 with probability p and 0 otherwise
   if (x == 1)
       w(t+1) = w(t) + b;               % If x = 1, then wealth increases by b
   else
       w(t+1) = w(t) - b;               % If x = 0, then wealth decreases by b
   end

   t = t + 1;                           % Increase time and repeat
end

% If the actual number of rounds is less than the maximum, the player went bankrupt
if t < max_t
    w = w(1:t+1);    % Truncate output
    h = 1;           % Set boolean variable
end

end
```

We just call the function with requested parameters and plot the results.

```
% Delete all variables and close figures
clear all
close all

% Simulation parameters
```

```
w_0 = 20;                           % Initial wealth
b = 1;                              % Bet value
max_t = 1000;                       % Maximum number of hands to simulate
p_vector = [0.25 0.50 0.75];        % Vector of probabilities of winning

figure;
hold on;
grid on;
xlabel('Hand');
ylabel('Wealth (in $)');
xlim([1 max_t]);
ylim([0 400]);

% p takes on a value from p_vector at each run of the loop
for p = p_vector
    [w, h] = gambling(w_0, b, p, max_t);      % Simulate gambling experiment
    plot(w, 'LineWidth', 2);                  % Plot wealth history
end

legend('p = 0.25', 'p = 0.5', 'p = 0.75');


%%% Export figure %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
set(gcf,'Color','w');
export_fig -q101 -pdf HW1_A.pdf
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Results are shown in Figure 1.

B Probability of ruin. When determining the number of experiments to run, it is important to keep in mind that more experiments lead to greater accuracy and that the only drawback of increasing the number of experiments is the increase in runtime. The following function will begin with $N = 100$ experiments, but will rerun with $N = 1000$ if too few players go bankrupt (which could indicate that the sample size was too small).

```
% Estimate the probability of ruin as a function of the winning probability 'p'
% and the initial wealth 'w_0'.
% The function makes sure at least 10 realizations go bankrupt or it increases
% the number of experiments. This tries to prevent inaccurate estimations
% based on too few samples.

function pr_ruin = HW1_B(p, w_0)
b = 1;      % Bet value
T = 100;    % Simulation will run for at most T iterations

% Run 100 experiments
N = 100;
nr_bankruptcy = 0;    % Initialization
for i=1:N
    % Simulate process
    [~, h] = gambling(w_0, b, p, T);

    if h ==1
        nr_bankruptcy = nr_bankruptcy + 1;
    end
end


if (nr_bankruptcy < 10)
    % Run 1000 experiments
    N = 1000;
    nr_bankruptcy = 0;    % Initialization
```

```

    for i=1:N
        % Simulate process
        [~, h] = gambling(w_0, b, p, T);

        if h ==1
            nr_bankruptcy = nr_bankruptcy + 1;
        end
    end
end

% Estimates the probability of the player going broke based on the
% number of experiments (N) and number of times the player went bankrupt
pr_ruin = nr_bankruptcy/N;

end
```

When called from the command window, we find the estimated probability to be

```
> prh(.55, 10)
ans =
    0.0890
```

Thus, for this experiment $\hat{B}_N(0.55, 10) = 0.0890$, i.e., the player is very unlikely to go bankrupt.

A much faster and concise version (with a little bit more MATLAB magic, so more involved as well) is shown below. Note that the following function does not call the `ssas-gambling` function. It is a slightly trickier way of calculating the same probability. If you want to know how it works but still have problems after consulting the MATLAB help, bring it up during the TA office hours.

```
% Estimate the probability of ruin as a function of the winning probability 'p'
% and the initial wealth 'w_0'.
% The function makes sure at least 10 realizations go bankrupt or it increases
% the number of experiments. This tries to prevent inaccurate estimations
% based on too few samples.

function pr_ruin = HW1_B_2(p, w_0)

b = 1;      % Bet value
T = 100;    % Simulation will run for at most T iterations
N = 1000;   % Number of realizations

steps = b * ( 2*random('bino',1,p,[T,N]) - 1 );
w = w_0 + cumsum(steps);
pr_ruin = sum( sum( w <= 0 ) > 0 )/N;

end
```

C Probability of ruin as a function of initial wealth. To determine the probability of going bankrupt as a function of wealth, we will use `HW1_B_2` with $p = 0.55$, vary w_0 between 1 and 20, then plot the results.

```
% Delete all variables and close figures
clear all
close all

p = 0.55;               % Probability of winning
pr_ruin = zeros(20,1);  % Initialize vector of the probabilities of ruin

for w_0 = 1:20
```

```
    pr_ruin(w_0) = HW1_B_2(p, w_0);
end

% Plot results
figure;
plot(1:20, pr_ruin, 'LineWidth', 2);
xlabel('Initial wealth');
ylabel ('Probability of bankrupcy');
grid;
xlim([1 20]);
ylim([0 1]);


%%% Export figure %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
set(gcf,'Color','w');
export_fig -q101 -pdf HW1_C.pdf
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The result is shown in Figure 2.

D Probability of ruin as a function of p. A plot of the results can be found in Figures 3.

We can clearly observe that for $p < 1/2$ the probability of bankruptcy approaches 1, i.e., the patron is very likely to lose all its money. From class, we know that this probability actually is one in the limit, i.e., when the gambler plays infinitely many hands. However, recall that we can only run finite simulations on our computers and that this is an approximation. In contrast, for $p > 1/2$, the probability of bankruptcy decreases to zero, i.e., with high probability the patron does not go broke. In other words, as the probability of winning p increases, the gambler can win large amounts of money (with increasing probability) by staying in the game long enough. When p is very close to 1/2, the probability of losing all money is extremely volatile. *Gambling lesson*: this game with $p < 1/2$ is extremely bad (bankruptcy is guaranteed).

```
% Delete all variables and close figures
clear all
close all

w_0 = 10;                              % Initial wealth
p_vector = 0.3:0.02:0.7;               % Vector of winning probabilities
pr_ruin = zeros(1,length(p_vector));   % Initialize output

for i = 1:length(p_vector)
    pr_ruin(i) = HW1_B_2(p_vector(i), w_0);
end

% Plot results
figure;
plot(p_vector, pr_ruin, 'LineWidth', 2);
xlabel('Winning probability (p)');
ylabel('Probability of bankrupcy');
grid;
xlim([0 1]);
ylim([0 1]);


%%% Export figure %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
set(gcf,'Color','w');
export_fig -q101 -pdf HW1_D.pdf
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

E Time to ruin. The estimated time to ruin obtain from the code are:

- for $w_0 = 10$:

  ```
  average T_0 = 52.590
  ```

- for $w_0 = 20$:

  ```
  average T_0 = 104.758
  ```

The resulting distributions are shown in Figures 4 and 5.

```
% Delete all variables and close figures
clear all
close all

p = 0.4;        % Probability of winning
w_0 = 10;       % Initial wealth
% w_0 = 20;       % Initial wealth
b = 1;          % Bet value
max_t = 1000;   % Maximum number of hands to simulate
N = 1000;       % Number of experiments

T_to_ruin = zeros(N,1); % Initialize output

for i=1:N
    [w, ~] = gambling(w_0, b, p, max_t);
    T_to_ruin(i) = length(w);
end

% Average time to bankruptcy
average_T_to_ruin = sum(T_to_ruin)/N;
fprintf('Average time to ruin = %.3f\n', average_T_to_ruin);

% Plot results
figure;
[n, xout] = hist(T_to_ruin, 25);
bar(xout, n);
xlabel('Time to ruin');
ylabel('Count');
grid;


%%% Export figure %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
set(gcf,'Color','w');
export_fig -q101 -pdf -append HW1_E.pdf
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

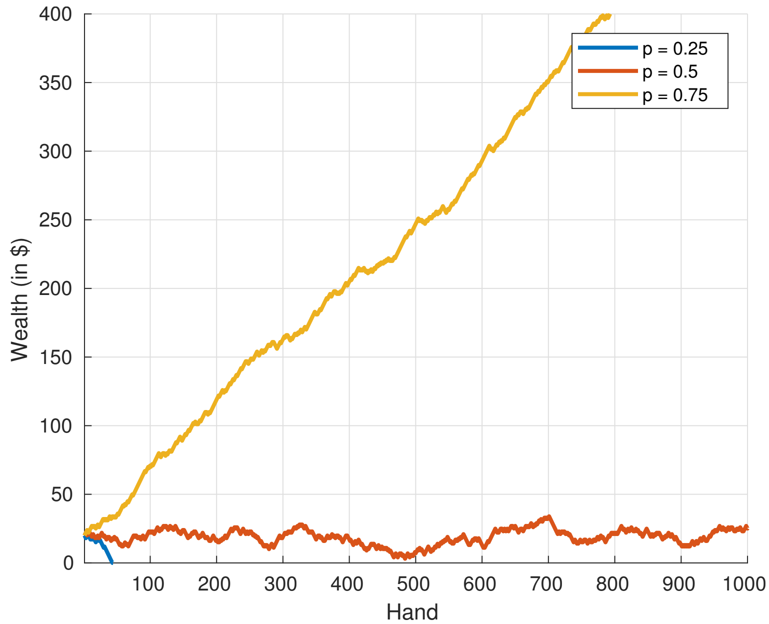

Figure 1: Simulated processes for $w_0 = 20$ and $p = 0.25$, $p = 0.5$, and $p = 0.75$ (Part A)

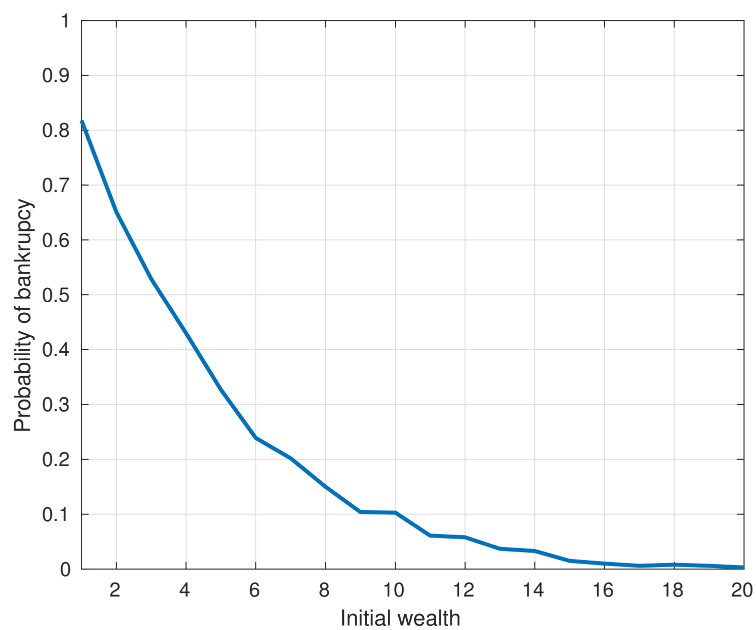

Figure 2: Probability of ruin as a function of the initial wealth for $p = 0.55$ (Part C)

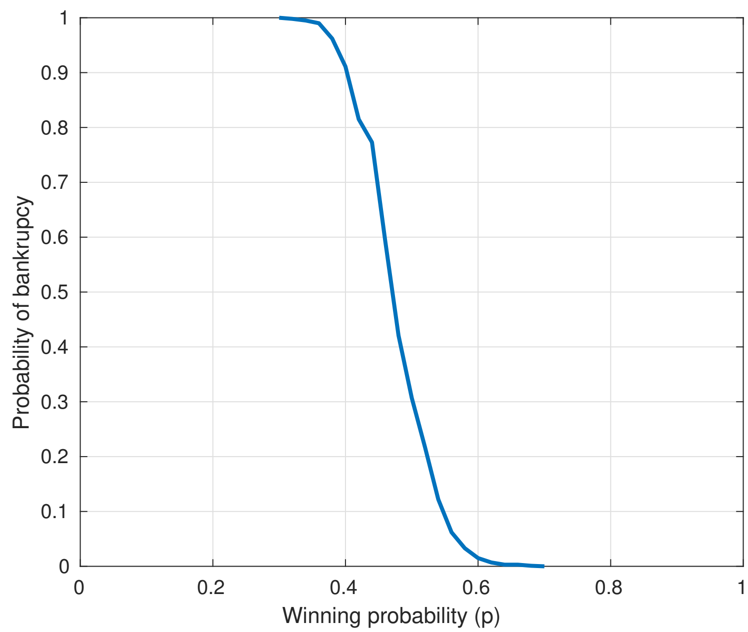

Figure 3: Probability of bankruptcy as a function of the winning probability for $w_0 = 10$ (Part D)

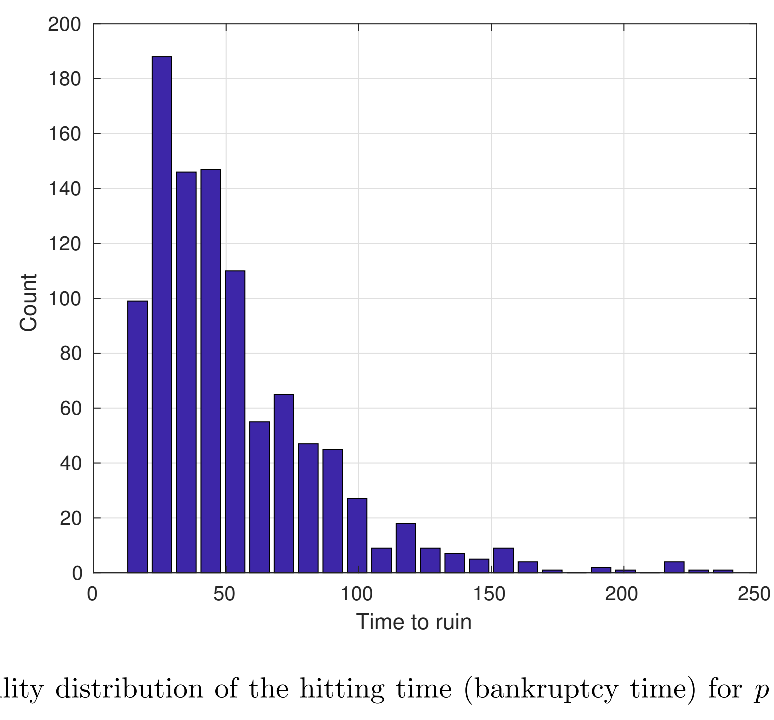

Figure 4: Probability distribution of the hitting time (bankruptcy time) for $p = 0.4$ and $w_0 = 10$ (part E)

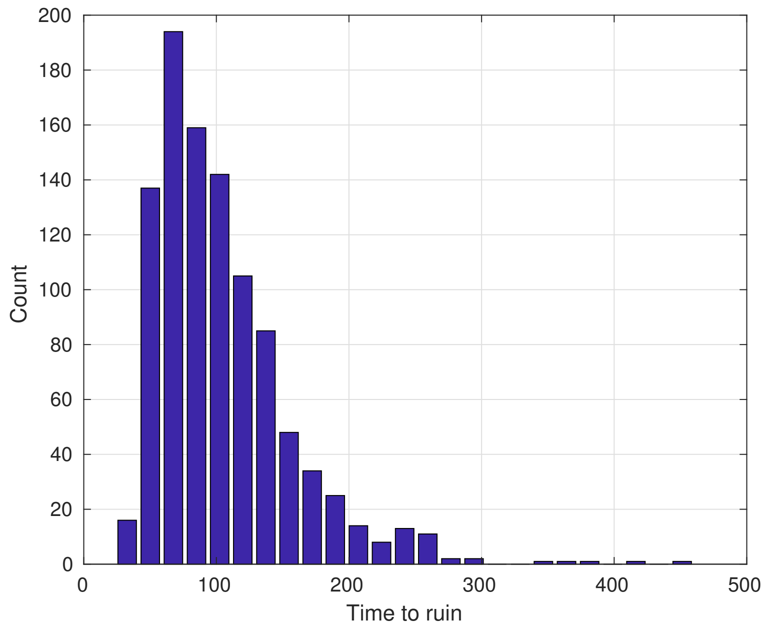

Figure 5: Probability distribution of the hitting time (bankruptcy time) for $p = 0.4$ and $w_0 = 20$ (part E)