Absorbing States and z-Transforms

ECE 341: Random Processes

Lecture #21

note: All lecture notes, homework sets, and solutions are posted on www.BisonAcademy.com

Absorbing States

Markov chains solve problems of the form

$$x(k+1) = A\ x(k) \qquad\qquad x(k=0) = X_0$$

In the case of three people tossing a ball in our last lecture, the ball keeps moving around and never ends up anywhere in particular. In some cases, there is a definite end point.

Example: Best of 5 Games

$$X(k+1) = \begin{bmatrix} 1 & 0.7 & 0 & 0 & 0 \\ 0 & 0 & 0.7 & 0 & 0 \\ 0 & 0.3 & 0 & 0.7 & 0 \\ 0 & 0 & 0.3 & 0 & 0 \\ 0 & 0 & 0 & 0.3 & 1 \end{bmatrix} X(k) \qquad X = \begin{bmatrix} \text{up 2 games (player A wins)} \\ \text{up 1 game} \\ \text{tied} \\ \text{down 1 game} \\ \text{down 2 games (player B wins)} \end{bmatrix}$$

Here, if you encounter state 1 or 5, the game ends.

- This is denoted in the state-transition matrix with a 1.00 in a row
- This is called an absorbing state.

If you have an absorbing state, as time goes to infinity you will always wind up there. This system has two absorbing states (player A wins and player B wins). The value of X determines the probability of getting to each of these states.

Find the steady-state solution

$$x(k+1) = Ax(k) = x(k)$$

$$(A - I)x(k) = 0$$

This doesn't help in this case due to the absorbing states. If you try to solve, you get

$$\begin{bmatrix} 0 & 0.7 & 0 & 0 & 0 \\ 0 & -1 & 0.7 & 0 & 0 \\ 0 & 0.3 & -1 & 0.7 & 0 \\ 0 & 0 & 0.3 & -1 & 0 \\ 0 & 0 & 0 & 0.3 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \end{bmatrix} = 0 \qquad X(\infty) = \begin{bmatrix} a \\ 0 \\ 0 \\ 0 \\ e \end{bmatrix}$$

result: someone wins

Option 2: Eigenvectors.

The eigenvalues and eigenvectors tell you

- How the system behaves (eigenvalues) and
- What behaves that way.

```
[M,V] = eig(A)

     1.0000          0   -0.8033    0.2846   -0.5384
          0          0    0.4039   -0.6701    0.7692
          0          0    0.3739    0.6203   -0.0000
          0          0    0.1731   -0.2872   -0.3297
          0     1.0000   -0.1476    0.0523    0.0989

V:   1.0000     1.0000    0.6481   -0.6481         0
```

The eigenvectors tell you that eventually,

- A wins (first eigenvector), or
- B wins (second eivenvector).

Option 3: Play the game a large number of times (100 times).

In Matlab:

```
A = [1,0,0,0,0;0.7,0,0.3,0,0;0,0.7,0,0.3,0;0,0,0.7,0,0.3;0,0,0,0,1]'

    1.0000    0.7000         0         0         0
         0         0    0.7000         0         0
         0    0.3000         0    0.7000         0
         0         0    0.3000         0         0
         0         0         0    0.3000    1.0000

A^100

    1.0000    0.9534    0.8448    0.5914         0
         0         0    0.0000         0         0
         0    0.0000         0    0.0000         0
         0         0    0.0000         0         0
         0    0.0466    0.1552    0.4086    1.0000
```

A^100 tells you the probability of

- A winning (first row) and
- B winning (last row)

The columns tell you the probability if you offer odds:

- Column 1: Player A starts with a +2 game advantage (player A always wins)
- Column2: Player A starts with a +1 game advantage (A wins 95.34% of the time)
- Column 3: Player A starts with a +0 game advantage (A wins 84.48% of the time)
- Column 4: Player B starts with a +1 game advantage (A wins 59.14% of the time)
- Column 5: Player B starts with a +2 game advantage (B always wins)

```
Odds   +2           +1           +0           -1           -2
     1.0000       0.9534       0.8448       0.5914            0     A wins
          0            0       0.0000            0            0
          0       0.0000            0       0.0000            0
          0            0       0.0000            0            0
          0       0.0466       0.1552       0.4086       1.0000     B wins
```

Good Money After Bad

- If you start losing, keep gambling to recoup your losses
- This tends to result in you getting further behind
- You're risking money you have (good money) to recoup money you lost (bad money)

For example, suppose you play a game of chance.

- 53% of the time you win and earn \$1.
- 47% of the time you lose and lose \$1.

Keep playing until you are up \$10

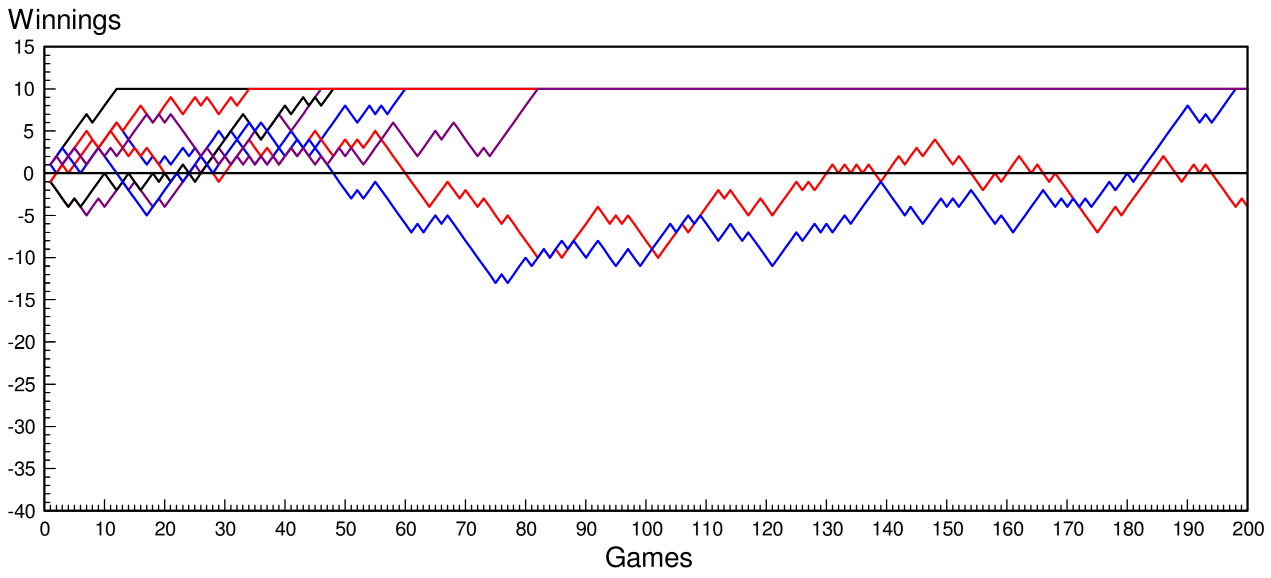
- Absorbing state

In theory, you always up \$10

Monte-Carlo Simulation

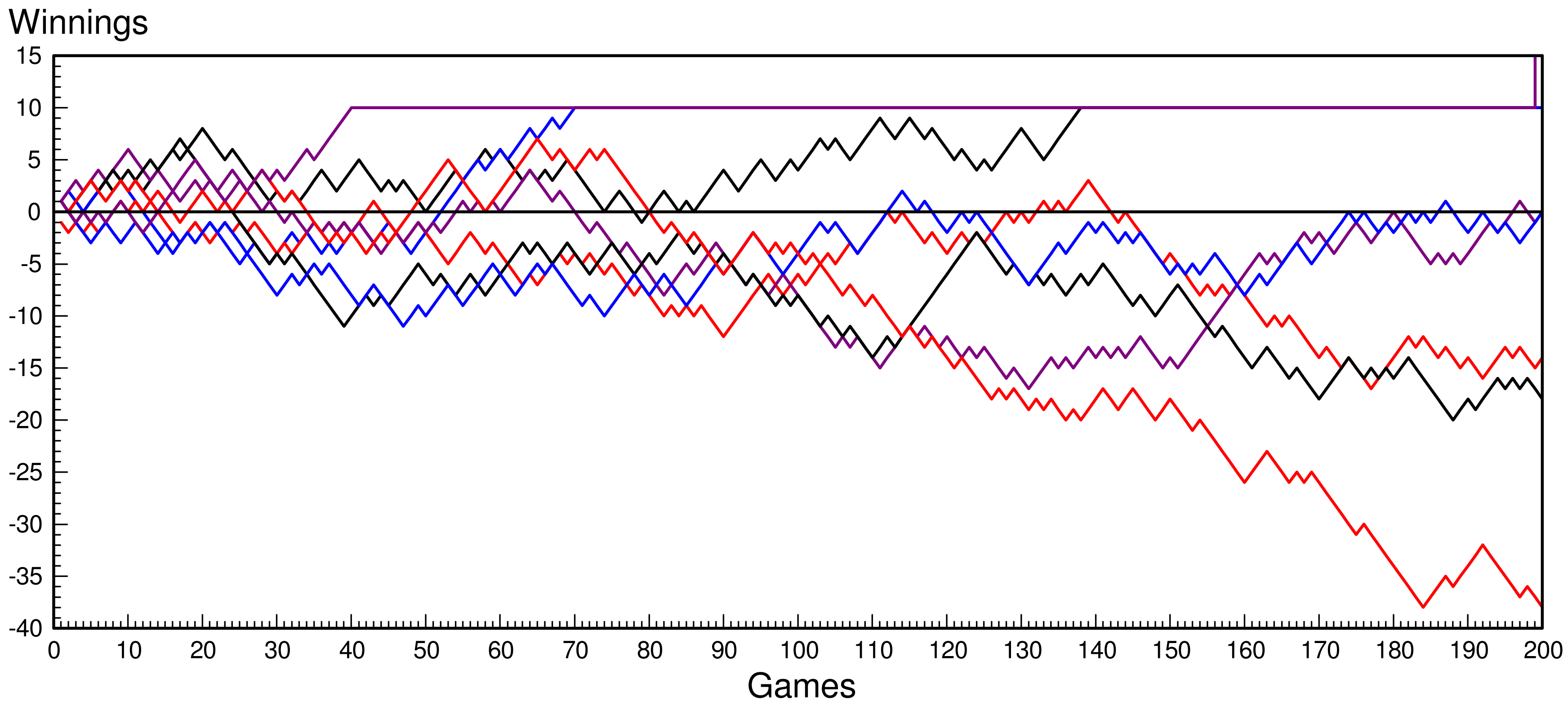
- Keep playing until you are up $10.

```
% game of chance

X = 0;   % winnings
n = 0;   % number of games

while (X < 10)
    n = n + 1;
    if (rand < 0.53)
        X = X + 1;
    else
        X = X - 1;
    end
    disp([n,X]);
end
```



Change the problem

- p(winning) = 47%



Winnings after n games with a 47% chance of winning any given game.

Sometimes you end up at the absorbing state (+$10)

Sometimes you keep getting further and further behind

- Good money after bad)

The creates a conflict:

- If you only have one absorbing state, you should always wind up at that absorbing state.
- With Monte Carlo simulations, this usually happens, but not always.

A better model would be to add a second absorbing state:

- Once you are up \$10, you quit and collect your winnings
- Once you are down \$100, the house no longer accepts your money.

Monte-Carlo Simulation: In 10,000 games

- 6,647 times you're up \$10
- 3,353 times you're down \$100

If you're losing, it's best to cut your losses and walk away.

- It could be you're losing because you're just not that good
- Your odds of winning are actually 47%, not 53%

z-Transforms

Suppose you want to determine the probability of player A winning after k games. This is where z-transforms shine.

z-Transforms designed to determine the time response of a discrete-time system

$$X(k+1) = AX(k) + BU(k)$$

$$Y = CX(k)$$

If U(k) is an impulse function, you get the impulse response:

$$zX = AX + B$$

$$Y = CX$$

This is what we want with Markov chains, only

- B is the initial condition: X(0)
- C tells you which state you want to look at.

For example, for the problem of winning by 2 games,

```
A =

    1.0000    0.7000         0         0         0
         0         0    0.7000         0         0
         0    0.3000         0    0.7000         0
         0         0    0.3000         0         0
         0         0         0    0.3000    1.0000

X0 = [0;0;1;0;0]

     0
     0
     1
     0
     0

C = [1,0,0,0,0]     % A wins

C =     1     0     0     0     0
```

You can now find the Y(z) (or the impulse response)

- Multiply by z (as per last lecture)

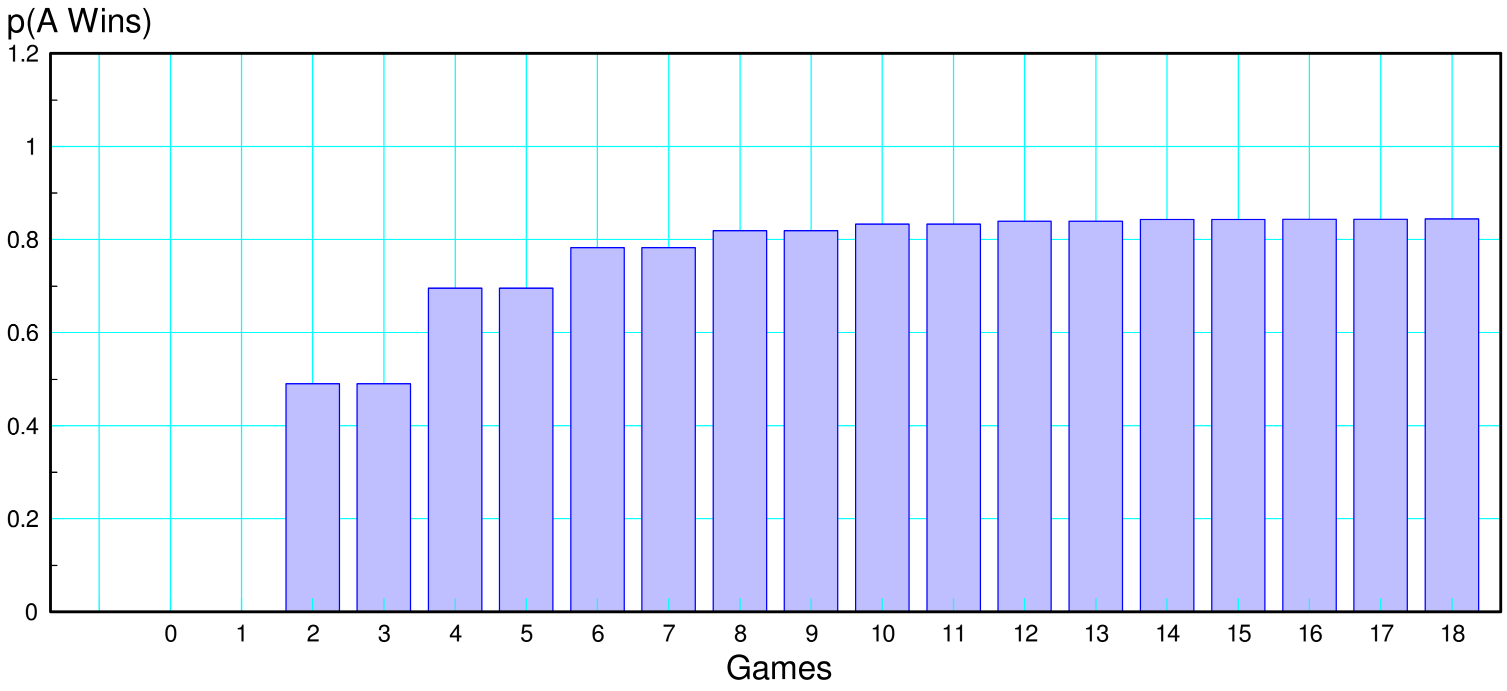
```
G  = ss(A,X0,C,0,1);

tf(G)

                  0.49 z
------------------------------------------
z^4 - z^3 - 0.42 z^2 + 0.42 z + 1.821e-018

sampling time (seconds): 1

zpk(G)

           0.49 z
-----------------------------
z (z-1) (z-0.6481) (z+0.6481)

Sampling time (seconds): 1
```

This tells you that

$$Y(z) = \left(\frac{0.49z}{(z-1)(z-0.6481)(z+0.6481)} \right)$$

The time response is from the *impulse* function

```
y = impulse(G)
```



You can also find the explicit function for y(k) using z-transfoms.

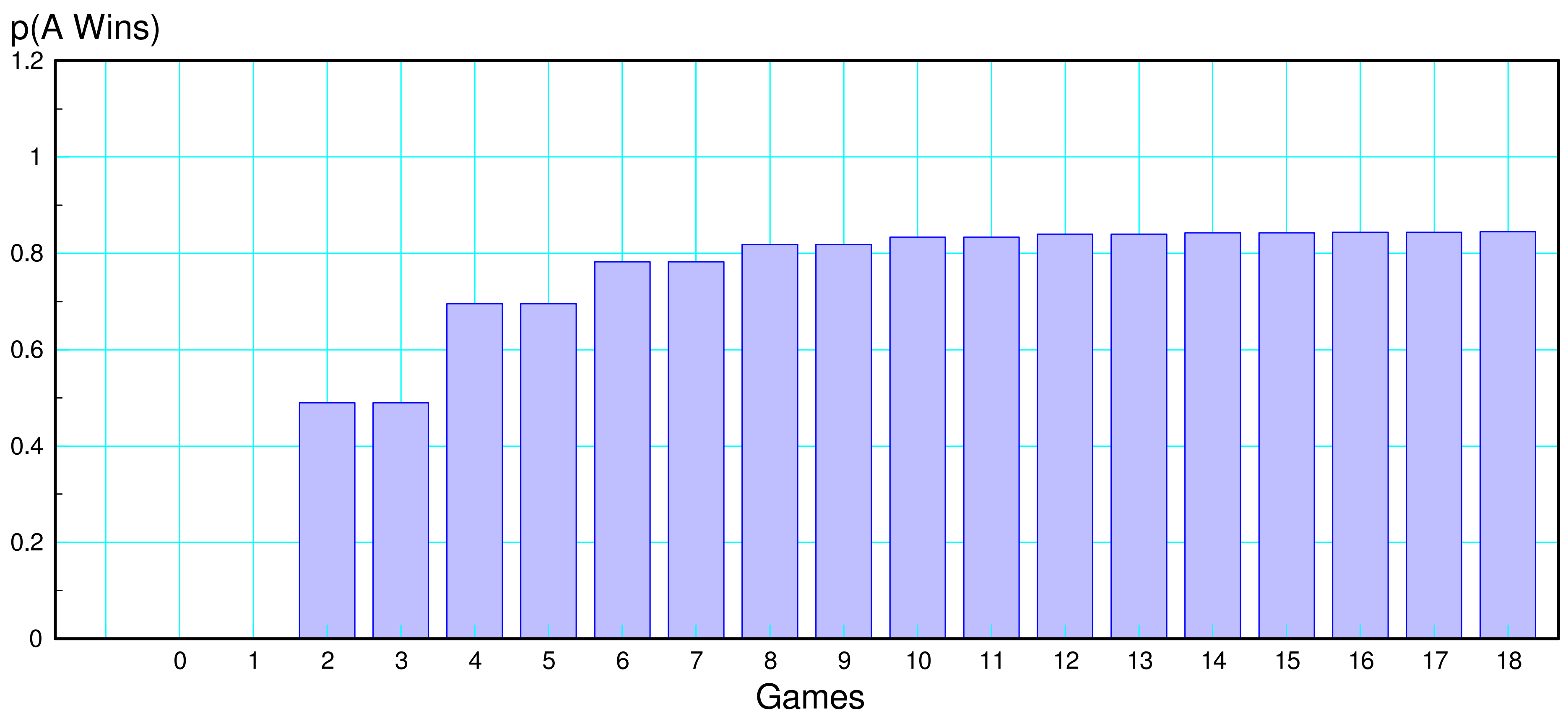
$$Y(z) = \left(\frac{0.49z}{(z-1)(z-0.6481)(z+0.6481)}\right)$$

Pull ou a z and do partial fractions

$$Y = \left(\left(\frac{0.8449}{z-1}\right) + \left(\frac{-1.0742}{z-0.6481}\right) + \left(\frac{0.2294}{z+0.6481}\right)\right) z$$

Take the inverse z-transform

$$y(k) = \left(0.8449 - 1.0742\ (0.6481)^k + 0.2294\ (-0.6481)^k\right)\ u(k)$$


p(A Wins)
1.2
1
0.8
0.6
0.4
0.2
0
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
Games