

# Project III — Expectation (Preliminary Version)\*

Peter Brinkmann

The project assumes that you have installed IPROM and that you have worked through the first three labs. In particular, you should be able to run simulations as well as change the parameters of simulations, and you should have an idea how to deal with vector-valued data and memory blocks.

## 1 The Game

Consider the following game: You flip a fair coin 100 times. Every time the coin comes up heads, you win \$1, and every time the coin comes up tails, you lose \$1. Throughout the game, you keep track of the number of steps after which you were ahead, i.e., steps after which you had won more than you had lost. This defines a random variable  $X$  taking values in the set  $\{0, \dots, 100\}$ .

We now want to develop an understanding of the probability mass function of  $X$ , i.e., we want to understand the probability  $P\{X = i\}$ , for  $i = 0, \dots, 100$ .  $P\{X = i\}$  is the probability of being ahead after  $i$  steps. This is hard to compute analytically, but IPROM simulations can give us numerical approximations of the exact results.

**Problem 1.** Before you continue, take a moment and think about what you would expect this probability mass function to look like. Specifically, which number between 0 and 100 do you expect to be the most likely, which should be the least likely? Do you have a guess regarding  $E(X)$ ?

---

\*Copyright © 2003, Peter Brinkmann ([brinkman@math.uiuc.edu](mailto:brinkman@math.uiuc.edu)). Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is available at <http://www.fsf.org/copyleft/fdl.html>.

Jot down your thoughts on this question. There is room for notes on the last page.

## 2 Simulating the game

The goal of this section is to build an IPROM model that simulates our game and computes the values we are interested in. We will get there by way of several intermediate steps. You may want to take another look at Lab II before you proceed.

**Problem 2.** Open the file `blank.mdl` and save it under a suitable new name. Think about the blocks you'll need to simulate the game. You'll certainly need a Memory block that'll remember your current balance, as well as a Coins block that'll toss a coin for you. How can you compute the new balance as a result of the most recent coin toss?

Copy all the blocks you need from the IPROM library, and connect them appropriately. In order to see whether your model is working correctly, you can copy a Bar Graph block from the library and have it display your current account balance. When you've completed your model, run it a few times in order to see whether the resulting bar graph is what it should be. Chances are that you'll have to go back and debug your model a few times before it works correctly.

Hint: The Coins block returns the value 0 for tails and 1 for heads. If  $x$  equals 0 or 1, then  $2x - 1$  will equal -1 or 1. This will help you compute the new account balance after each step.

When you're confident that your model does what it's supposed to do, print it and submit the printout. If you have trouble getting the simulation to work, you may find the Simulink debugger helpful (see Lab I).

**Problem 3.** Now, modify your model so that it counts the steps after which you're ahead. You may want to use the Conditional Counter block from the IPROM library. This conditional counter only counts those steps after which the value at its top input port is nonzero.

In order to see whether the modified model is working correctly, you can feed the output of the counter into a display block. Run the model a few times to see whether it works, and debug if necessary.

When the modification works, print the new model and submit the printout.

### 3 Evaluating the outcome

**Problem 4.** In order to repeatedly simulate our game, we need to be able to run our simulation within another simulation. Copy an Output block from the IPROM library and connect it to the output of the counter. This allows other simulations to run our new simulation and see its output.

Now, open the file `puzzler9.mdl` that you already know from Lab II and save it under a new name. It happens to do precisely what we want to do here: It runs another simulation and displays its output as a histogram. It also computes the average value of the output of the interior model. In order to have it use the model that you built in the previous section, double-click on the Run Model block and replace the name `odd` by the name of your new model (without the `.mdl` extension).

Run this model a few times in order to see whether it works as expected. You may want to increase the number of iterations by changing the stop time of the model (see Lab I) to, say, 2000 or more, depending on how patient you are.

Print the histogram and submit the printout.

**Problem 5.** Take a look at the resulting average as well as the histogram. Answer the following questions.

1. Does the average value agree with your expectations?
2. Describe the histogram. Does it agree with your expectations, i.e., do you see the tallest spikes at the values that you originally expected to be the most likely?
3. How do the histogram and the average value fit together?
4. Can you think of an intuitive explanation for the histogram you are seeing?

There is room for notes on the last page.

NAME:

Notes for Problem 1:

Notes for Problem 5: