CS1133, Spr 2018, HW 5 (Lect 9, Pts=40+40)

Due February 27, 2018 (before 11:00 pm)

Problem 1 [40 pts]

We want to use Monte Carlo simulation to investigate another property of a hypercube in n dimensional space, when n is large. We live in a three-dimensional space and so smoe of these properties are weird to us.

An n-dimensional hypercube is a "cube" in an n-dimensional space. For $n = 1$, it is a line segment. For $n = 2$, it is a square, and it is a unit cube for $n = 3$. This idea can be generalized to 4 and higher dimensions.

Here is a good way to describe such a hypercube in any n-dimensional space: any arbitrary point within the hypercube must have a total of n independent coordinates. The coordinates are given by $(x_1, x_2, \ldots, x_n)$ where each of its n components has a value lying within the range $[-1, 1]$. Thus the length of each side of the hypercube is given by 2.

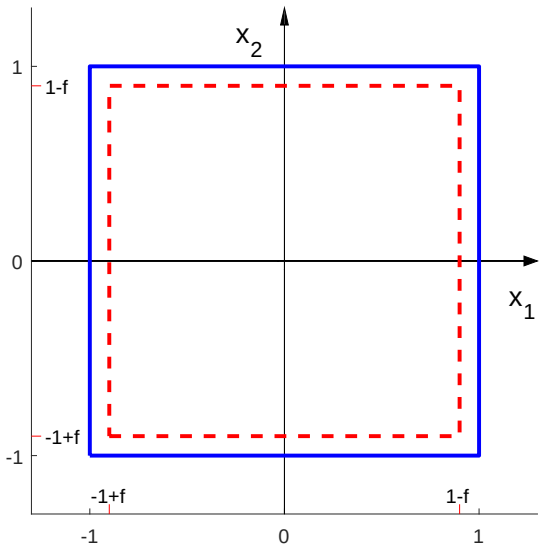
In the case of one-dimension, we have $n = 1$. Thus each point within the hypercube has only one coordinate, x, with x ranging from -1 to 1. The hypercube is simply a straight line extending from -1 to 1, with a length of 2.

In the case of two-dimension, we have $n = 2$. The hypercube is a square centered at the origin, and has a length of 2 and boundaries at $x_1 = \pm 1$, $x_2 = \pm 1$.

In three-dimension, a hypercube is the usual cube centered at the origin, with boundaries at $x_1 = \pm 1$, $x_2 = \pm 1$, and $x_3 = \pm 1$.

We want to use simulation to find the fraction of all the points within an n-dimensional hypercube that lie within a distance of f from any one of its boundaries. Here we can take $f = 0.01$. You must write your program in a flexible way so that it can be used to find the answer for any dimension, n, that the user enters during runtime from the keyboard. Perform a single simulation employing around $100,000$ n-dimensional points.

The following drawing illustrates the problem in the special case of two-dimension. The value of f used was $f = 0.1$ (10 times what it is supposed to be for illustration purposes). Your program must work for any given positive integer dimensions.



Actually because of the symmetry of the problem, we do not have to work with the entire hypercube. All we need is to concentrate on the positive "quadrant" of the hypercube where the coordinates of each point are all positive.

Here is an example of one output displayed by the program:

```
For a 200-dimensional hypercube, the fraction is: 0.8641
```

Your program should produce similar results in this format.

First test your program by setting $n = 1$. What answer do you get? Is the answer what you expect? Then try $n = 2, 5, 20, 100, 300, 600$. Do you see a trend? What does the trend tell you above high-dimensional hypercubic spaces?

Problem 2 [30 pts]

We are interested here on dices that each has n faces. We assume that when a die is thrown or rolled, it comes to rest showing on its upper surface a random integer from 1 to n, each value being equally likely. Other than the ordinary cubic shape, that has $n = 6$ faces, dice having different n have been used recently in war games, role-playing games, and trading-card games.

Your program must allow the user to choose dice having n given by $4, 6, 8, 10, 12, 20, 24$, or 30. Use Matlabs `menu` function together with vector indexing to allow the program to select the value of n (specifying the type of dice to use).

Your program can use an assignment statement to specify the number of dice, m, to use in a given trial.

Here we want to perform computer simulation to find the probability that, in a game involving throwing simultaneously m n-sided dice, there are more prime numbers than half of the number of even values, and at the same time, no more than half of the values are odd. To get good reliable results, perform a simulation using a fairly large number of trials (about serval thousands).

The function `isprime` can be used for any given matrix, say `MTX`, to detect if its elements are prime numbers (an integer greater than 1 that is not divisible by any number except 1 and itself) or not. The result for `isprime(MTX)` is a logical matrix the same size as `MTX`. The resulting element of the logical matrix is true for each element in `MTX` that is a prime number; otherwise, it is false.

Here is an example of the output display that your program needs to produce:

```
7 8-sided dices are thrown.
Computed probability for meeting the condition is: 0.30657
```

Your program should produce similar results in this format.