CBE 5790 Modeling and Simulation (Autumn 2018)

Homework [P1, P2]

Deadline for uploading programs to Carmen: Thu, Sep 20 at 2:20 PM
Deadline for completing Carmen quizzes: Thu, Sep 20 at 11:59 PM

Problem P1 – Pseudorandom number generator

a) Create your own function to generate pseudorandom numbers based on the example discussed in class. You may use Professor Rathman's code as a starting point, but your function will be much more versatile. Your function must be designed as follows:

	name	data type	default value	description
parameters	*size*	int or tuple of ints	None	Shape of output; if None, a single value is returned.
	method	string	'NR'	LCG algorithm to be used; accepted values are 'NR' or 'RANDU'. If you wish, you may also invent your own method and include it as well.
	seed	float	None	Seed to be used in LCG algorithm; if None, your program will generate a seed.
	returnSeed	Boolean	False	Specifies whether to return seed value as output.
out	Returns a single float or an ndarray of floats. If returnSeed is True then the seed is also returned as a second element of the output tuple.			

Your function must be named **p1** and a user should be able to call your function using positional arguments, keyword arguments, or a mixture of both. For example, calls that should work for your function include:

```
p1()
p1(5)
p1((3,20), 'RANDU')
p1(200, returnSeed = True)
p1(method = 'RANDU', size = (4, 4))
```

As discussed in class, the method used in my program for generating a seed was not particularly good; try to devise a better strategy for generating a seed in your program. Although there are websites such as www.random.org and qrng.anu.edu.au that claim to provide "true" random numbers, you can not use these in this assignment – not even for just getting a seed. You must generate your seed internally (e.g., no reliance on an internet connection).

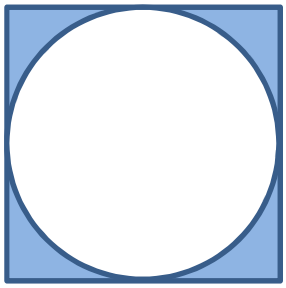
Important: This function must ONLY return the output random number(s) and the seed (if requested) and nothing else! No graphs, no prompts, no print() statements! Your function should "run silent" except for the outputs returned to the caller.

b) Write a Python script that analyzes the results of your random number function to see if you can discover any evidence that it does not generate decent random numbers. One way of doing this is to use the function to generate a large number of points that can be graphed in a 3-dimensional plot; if the generator is a good one, this plot should have no obvious patterns – the points should be randomly distributed throughout space. For example, a single call to your function such as p1((5000,3)) should return an ndarray with 5000 rows and 3 columns, which could then be plotted in a 3-dimensional scatter plot. Prepare such plots to compare results for the two LCG generators (NR and RANDU) and also NumPy's own uniform random number generator, np.random.random_sample(). **Hint:** rotate the plots in 3D to see if any patterns are observed from different perspectives. One of the quiz questions will ask you to write a short summary of your observations.

c) You should observe something odd with the RANDU generator in part (b); think about the consequences of using this generator to model stochastic processes. **Hint:** the problem with RANDU surfaces when the (x,y,z) coordinates for each point are obtained from 3 successive calls to the function. One of the quiz questions will be: What unexpected problems might users have if they relied on this function to generate random numbers?

d) To test the method you used to generate a seed value, you could do something similar to the approach in (b). You could, for example, populate the 5000 × 3 matrix with 15,000 seed values. Use this approach – or some other approach that you think would be appropriate – to test your algorithm for calculating seed values. You will be asked to answer this question in the quiz.

Problem P2 – The π *hole*



Imagine a square dartboard with an inscribed circle and a rather unskilled dart thrower who manages to hit the dartboard every time but at random locations. After a large number of darts have been thrown, if their distribution is random and uniform then the value of π can be estimated from calculating the proportion of darts that landed within the circle. (Be sure you can explain why this is true!)

Simulate this experiment using your random number generator from problem P1. Your function must be named **p2**. You function should allow a user to specify the total number of throws and the LCG method (NR, RANDU, …) used by the generator. A figure should be created illustrating the dartboard and location of the throws. The estimated value of π should be an output variable of your function (and it would also be very nice to display it in the figure as well). Your function **p2** must be designed as follows:

	name	data type	default value	description
parameters	*nThrows*	int	200	Number of darts to be thrown
	method	string	'NR'	LCG algorithm to be used; accepted values are 'NR' or 'RANDU'. If you wish, you may also invent your own method and include it as well.
out	Returns estimate for π			

Note that the problem with RANDU can also be observed in the dartboard simulation if the x and y coordinates of each point are generated by two consecutive calls to the random number generator.

How and what to submit: Upload a single py-file that contains both your **p1** and **p2** functions, and also complete the quizzes in Carmen. The name of your py-file should be a concatenation of your last name, your first name, and the problem number; e.g., **simpson_lisa_p1p2.py**. Your programs must be properly commented, including a docstring at the beginning of each function that provides information about what the function does and how to use it, creation and revision dates, your name, and acknowledgement of help you received from other people or other resources beyond course materials. For example:

```
def p1( . . . ):
    """
     Pseudorandom numbers uniformly distributed [0 1]
     Uses a linear congruential generator algorithm to
     generate a column vector of N pseudorandom numbers uniformly distributed
     from 0 to 1. Parameter values for the generator depend on the specified
     METHOD, a string variable that must be either 'NR' (Numerical Recipes)
     or 'RANDU' (the RANDU generator)

     James F. Rathman
     Created: 09/15/2015
     Revised: 09/14/2017
     People who helped me: Bryan Hobocienski, Edward Hughes,
                           Edna Krabappel and Nelson Muntz
     """
```