MITx CSE.0002x

Introduction to Computational Science and Engineering

Help sandipan_dey

Course Progress Dates Discussion MO Index

Course / 15 Fundamentals of Probability and Statisti... / 15.2 Fundamentals of Probabil...

‹ Previous ✓ ✓ ✓ ✓ ✓ Next ›

15.2.6 Example: Projectile with Uniform Distributions

Bookmark this page

MO2.1 MO2.3 MO2.13 MO2.14 MO2.15

In this example, we will consider the projectile problem with V_0 and θ_0 uniformally distributed for:

- $28\,\mathrm{m/s} < V_0 < 32\,\mathrm{m/s}$
- $25^\circ < \theta_0 < 35^\circ$

The Python implementation is shown in the code below.

```
import numpy as np
import matplotlib.pyplot as plt

def calc_xf(V0, th0, g):
    """
    Calculates the impact location (xf) of a projectile
    given the initial speed, angle, and gravity.

    Args:
        V0 (float or numpy array of floats): Initial speed
        th0 (loat or numpy array of floats): Initial angle
(degrees)
        g (float): gravity.

    Returns:
        xf as either a float or numpy array (if V0 and th0 are
numpy arrays)

    """
    xf = (V0**2)/g*np.sin(2*th0*np.pi/180.)
    return xf

def run_xfMC(V0lim, th0lim, g, N):
    """
    Perform a Monte Carlo simulation of sample size N
calculating xf
    the point of impact of a projectile on the ground with
initial
    speed V0 and initial angle th0 are uniformally distributed.

    Args:
        V0lim (tuple of floats): V0lim[0] gives minimum V0 and
        V0lim[1] gives maximum V0
        th0lim (tuple of floats): th0lim[0] gives minimum th0
and
        th0lim[1] gives maximum th0.  Note: th0lim will be in
degrees.
        g (float): gravity
        N (integer): size of sample to run Monte Carlo on

    Returns:
        xf (numpy array of floats): point of impact for all
instances in sample

    """
    rng = np.random.default_rng()
```

Discussions

All posts sorted by recent activity

```
    rng = np.random.default_rng()

    V0  = rng.uniform(V0lim[0],   V0lim[1], N)
    th0 = rng.uniform(th0lim[0], th0lim[1], N)

    xf = calc_xf(V0,th0,g)

    return xf


def calc_ptarget(xf, xftarget):
    """
    Calculate probability (fraction) of instances of xf which
    fall between xftarget[0] < xf <  xftarget[1]

    Args:
        xf (numpy array of floats): impact locations
        xftarget (tuple of floats): gives desired impact
location range

    Returns:
        Ptarget (float): estimated probability
```

‹ Previous

Next ›

```
    Ntarget =
np.count_nonzero(np.logical_and(xf>xftarget[0],xf<xftarget[1]))
```

edX

About
Affiliates
edX for Business
Open edX
Careers
News

Legal

Terms of Service & Honor Code
Privacy Policy
Accessibility Policy
Trademark Policy
Sitemap
Cookie Policy
Your Privacy Choices

Connect

Idea Hub
Contact Us
Help Center
Security
Media Kit

0.01