Solution : Homework 6

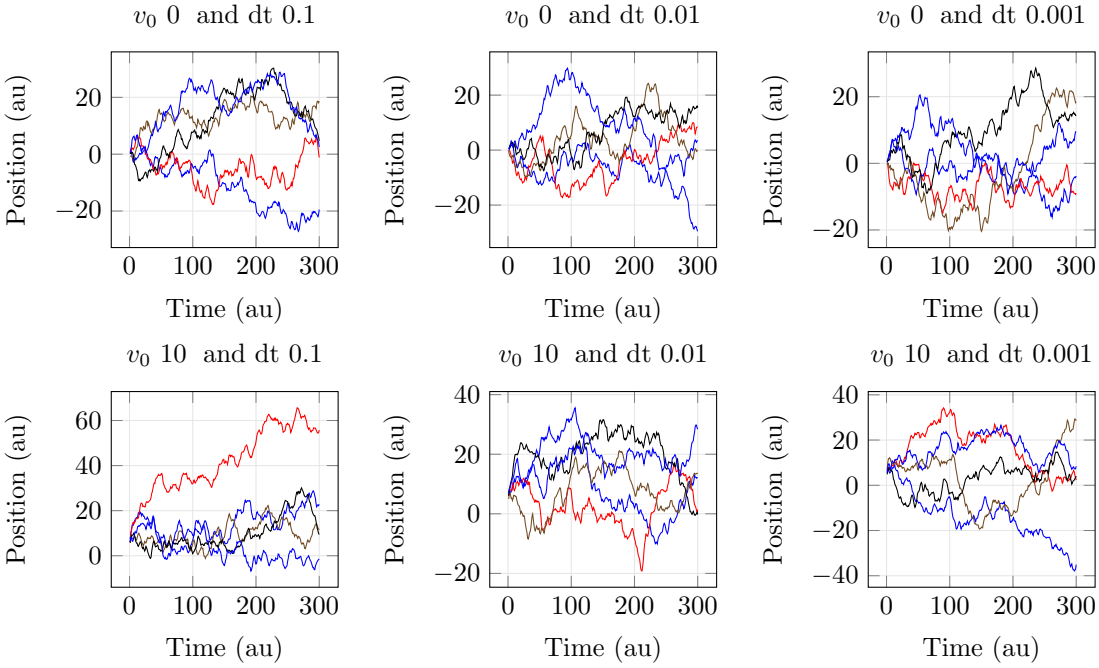
Dilawar Singh

March 29, 2017

The simulation is done in Haskell; Since Haskell is not a very mainstream language, code is in appendix for your reference only. It is available on demand. It can also be downloaded from github.

1 Solution

For each case we plot 5 sample trajectories.



Part B

Next, we plot the histogram if velocity. Is it possible to tell if all of them are same? By the look from it, they look the same. Usually one do statistical test (e.g. KS test) to verify.

Bonus

We use the same data to compute the diffusion coefficient using $(x_t - x_0)^2 = 2Dt$. After some time, the slope is getting less variable and seems to be getting constant.

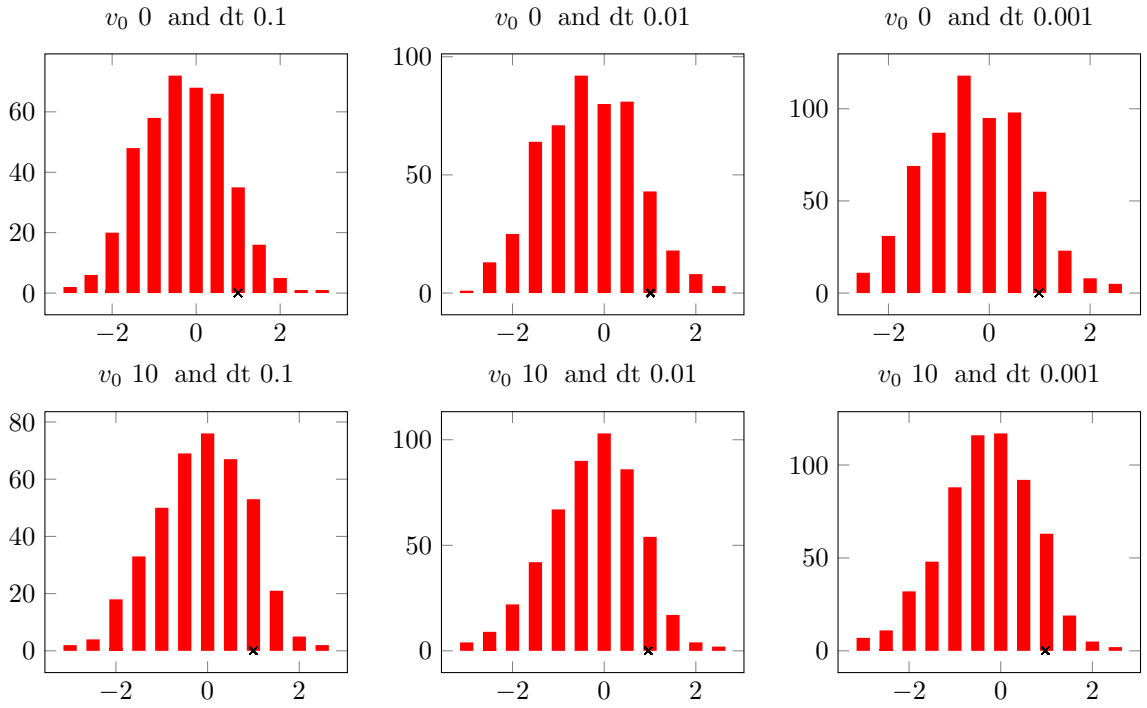
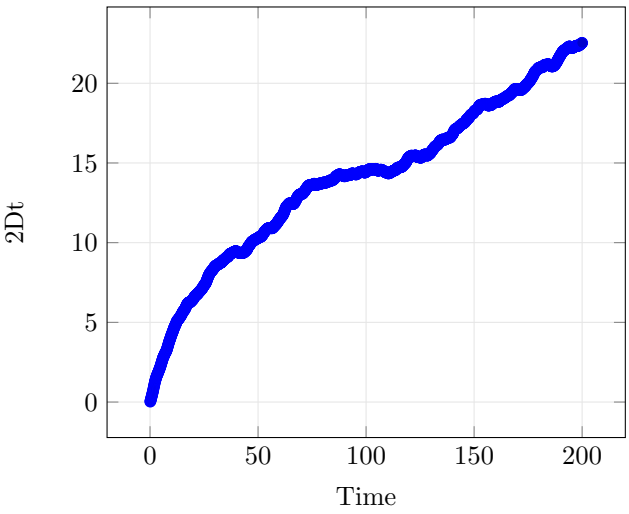


Figure 1: Histogram of velocity. The black cross is the root-mean-squared value of velocity as 100 units of simulation. The histogram look similar for given number of entries.



A Haskell based simulator

```
-- Solution to problem 6.

import Data.Random.Normal -- from normaldistribution
import System.Random
import Data.List
import System.Environment

-- We keep the variables here.
mByGamma = 1
ktbyM = 1


simulateUsingLangevian v0 steps dt = do
    g <- getStdGen
    let alphas = take steps $ normals' (0, 1.0) g
    return $ simulate alphas 0.0 v0 0.0 dt

-- Compute next v
nextV alpha v dt = v - (v/mByGamma*dt) + alpha * ((2*ktbyM/mByGamma)**0.5) *(dt ** 0.5 )

-- Simulate
simulate [ ] _ _ _ _ = [ ]
simulate (a:as) x0 v0 t0 dt = [t0+dt,nv,nx] : simulate as nx nv (t0+dt) dt
  where
    nv :: Double
    nv = nextV a v0 dt
    nx :: Double
    nx = x0 + nv * dt


-- Helper function. Write CSV file.
lineToStr :: [ Double ] -> String
lineToStr line =  intercalate "," $ map show line
writeCsv datafile xs = do
    let text = "time,velocity,position\n" ++ (unlines $ map lineToStr xs)
    writeFile datafile text

toDouble :: String -> Double
toDouble x = read x

main = do
    args <- getArgs
    -- Args are stop, step, v0 and outfile name.
    let (stop:step:velocity:filename:[]) = args
    let nSteps = floor $ (toDouble stop) / (toDouble step)
    let v0 = toDouble velocity
    trajectory <- simulateUsingLangevian v0 nSteps (toDouble step)
    -- Save trajectory to a CSV file
    writeCsv filename trajectory
    putStrLn $ "Wrote data to " ++ filename
```