Velocity of a Brownian particle

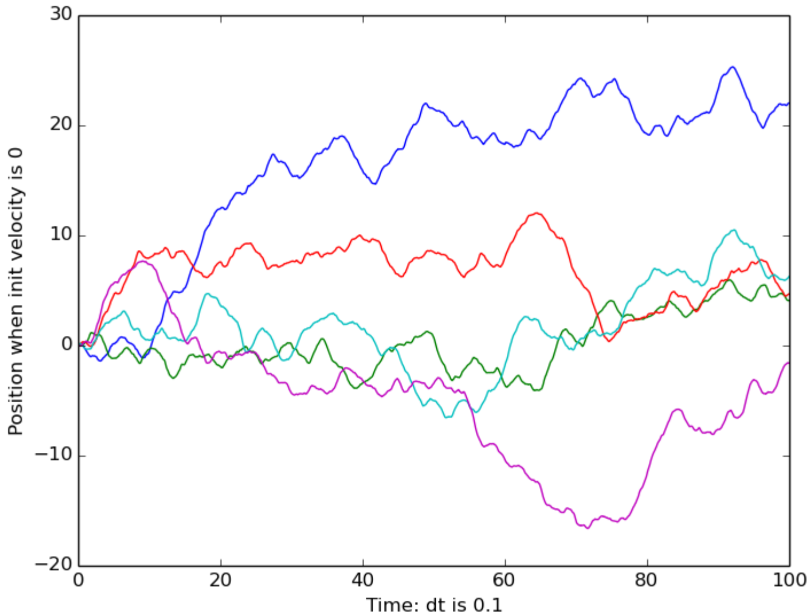
Dilawar Singh

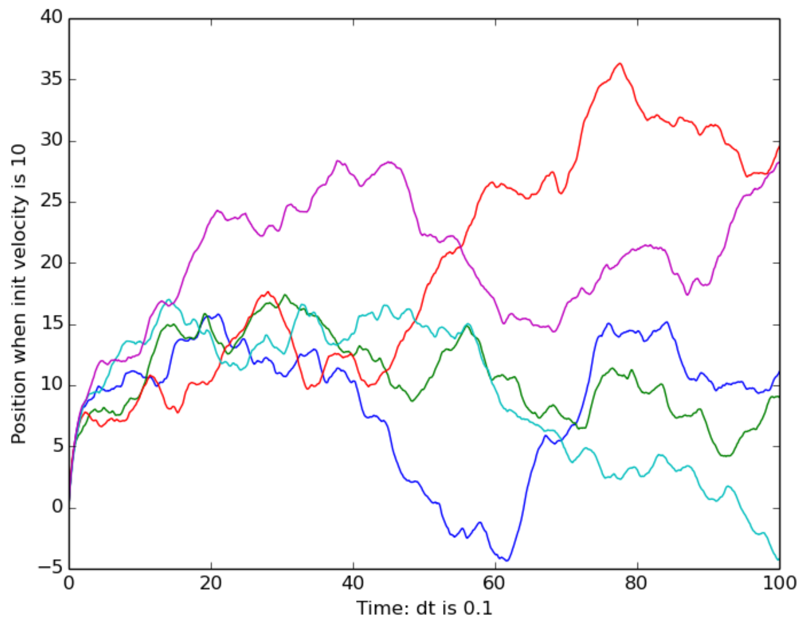
October 1, 2014

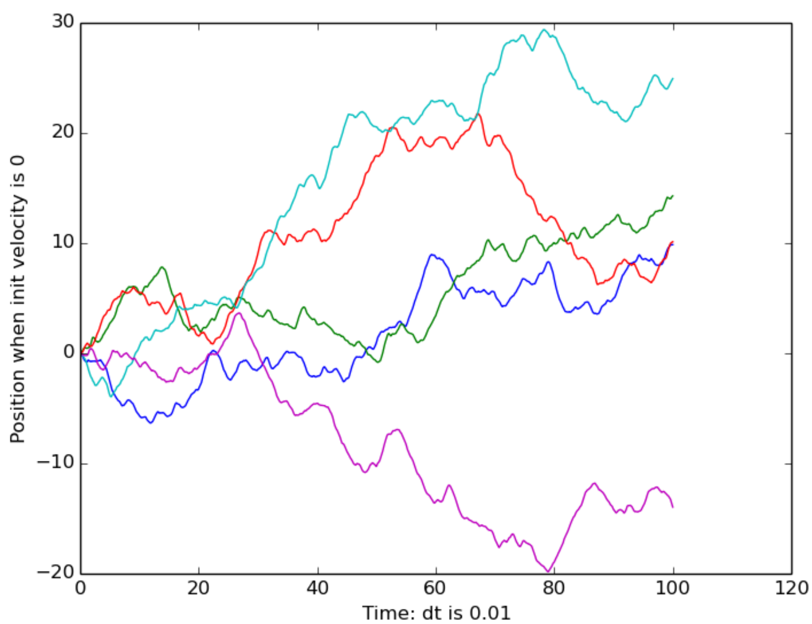
1. Equation of velocity

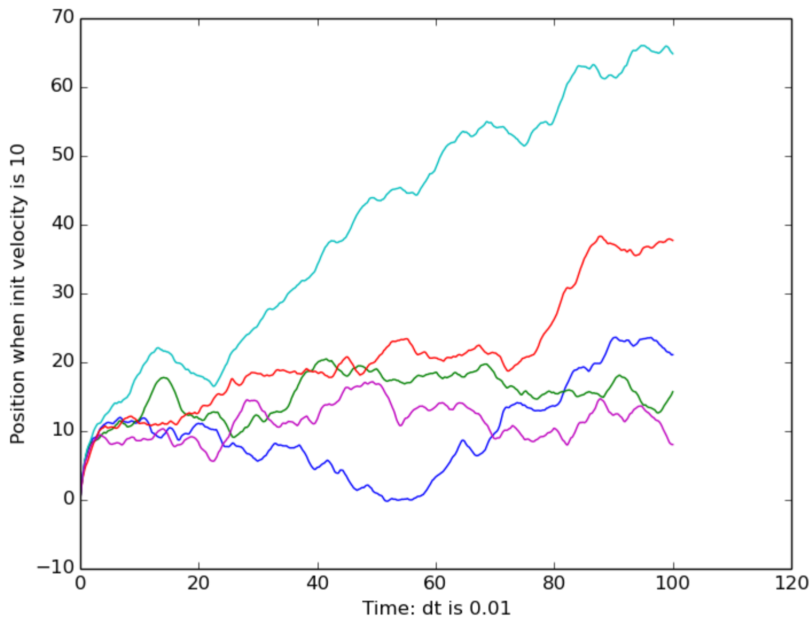
$$\Delta x = \frac{dx}{dt}\Delta t \Delta v = (-\Gamma v/m)\Delta t + \alpha\sqrt{2\Gamma kT/m^2}\sqrt{\Delta t}$$

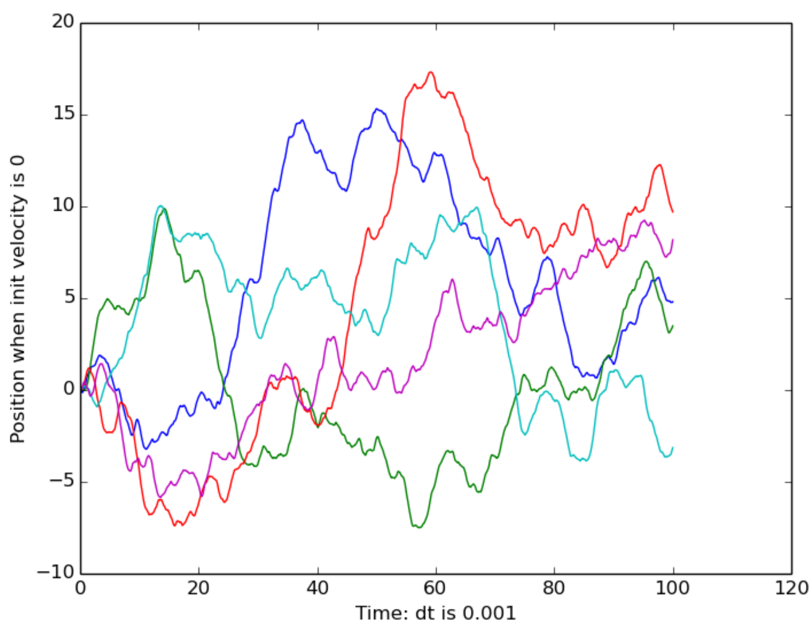
We simulated this equation and calculated the trajectories of randm walk. For initial velocities 0 and 10, and step size of 0.1,0.01,0.001; we calculated 5 trajectories for each case. Which are following.

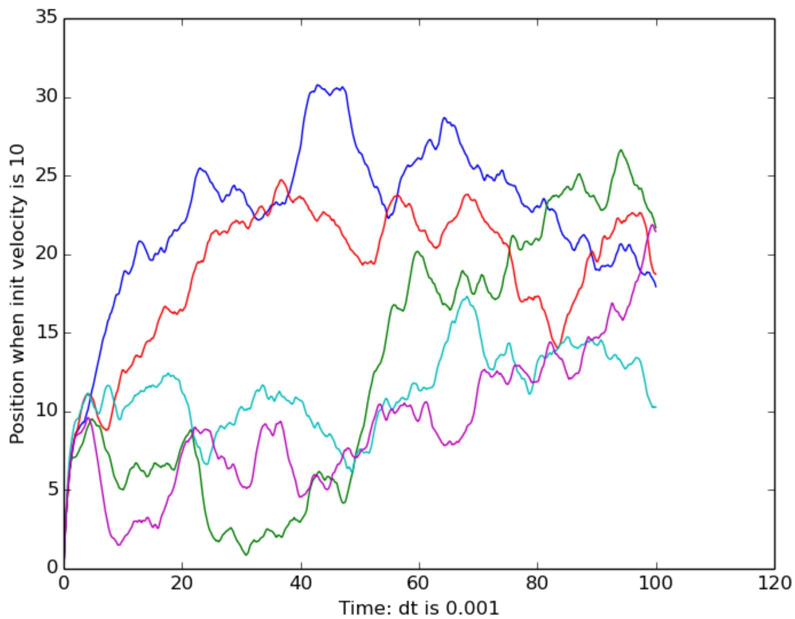



Position when init velocity is 10
Time: dt is 0.1
40
35
30
25
20
15
10
5
0
−5
0
20
40
60
80
100


Position when init velocity is 0
Time: dt is 0.01
30
20
10
0
−10
−20
0
20
40
60
80
100
120


Position when init velocity is 10
Time: dt is 0.01
70
60
50
40
30
20
10
0
−10
0
20
40
60
80
100
120


Position when init velocity is 0
Time: dt is 0.001
20
15
10
5
0
−5
−10
0
20
40
60
80
100
120



The mean and variances for these cases does not seem to converge to any particular values. The RMS values of velocities is in the last column of the following table.

samples	dt	initv	mean	variance	rms
5	0.001	10	-0.390393329705	0.43607357852	0.585292335301
5	0.01	0	-0.00203837453965	0.776564832395	0.776567507615
5	0.1	10	0.600189421336	0.680335015228	0.907239259748
5	0.1	0	0.420841728654	0.199954004038	0.465928496989
5	0.001	0	0.828840866831	0.9631447311	1.27067893489
5	0.01	10	0.155352789394	1.23659534669	1.24631558629

However, when 1000 trajectories were generated for each of these cases, both mean and variances seem to be converging to some value.

samples	dt	initv	mean	variance	rms
1000	0.001	10	-0.0460672892944	0.967197091056	0.968293554709
1000	0.01	0	-0.0227994089047	0.993544538457	0.993806099269
1000	0.1	10	-0.0181342038293	1.02198523926	1.02214611412
1000	0.1	0	0.0764892808579	1.05055051585	1.05333138017
1000	0.001	0	-0.0731812176636	1.00866496657	1.01131622424
1000	0.01	10	0.0172011145406	1.02220144662	1.02234616242

It would be interesting to see when and how fast mean and variance converges.

The bonus thingy

The value of diffusion coefficient in this case is $\frac{\Gamma k T}{m^2}$ which is 1.0. I am not getting it using the method described. The source code which I am using is attached below.

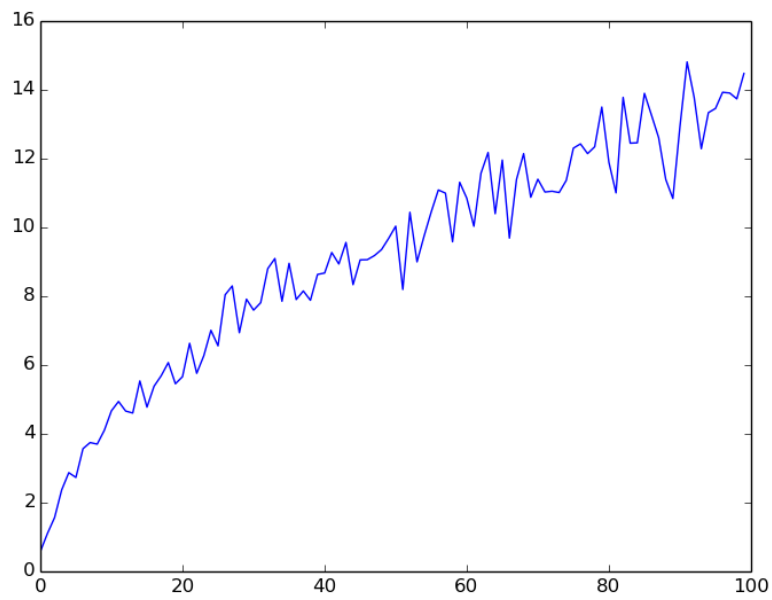


Figure 1: Diffusion constant i.e. slope of this curve is not 1.0

```
import numpy
import sys
from collections import defaultdict

class Brownian():

    """
    The equation is the following

    dv = (-v/a) dt + alpha * sqrt(2*b/a*dt)

    alpha is distributed with mean 0 and variance 1.
    """

    def __init__(self, dt, initV = 0.0, runtime = 100.0):
        self.dt =  dt
        self.steps = int(runtime/dt)
        self.dxs = numpy.zeros(self.steps)
        self.time = numpy.zeros(self.steps+1)
        self.pos = numpy.zeros(self.steps+1)
        self.a = 1.0
        self.b = 1.0
        self.alpha = numpy.random.normal(0.0, 1.0, self.steps)
        self.initV = initV
        self.finalV = 0.0
        self.runtime = runtime

    def reset(self):
        self.__init__(self.dt, self.initV, self.runtime)
```

```
    def computeV(self, v, alpha):
        dv = (- (v / self.a) * self.dt) + (alpha * numpy.sqrt(2.0 * self.b / self.a * self.dt))
        return (v + dv)

    def solve(self):
        t = 0.0
        v = self.initV
        for i, a in enumerate(self.alpha):
            t = t + self.dt
            vnew = self.computeV(v, a)
            self.time[i+1] = t
            self.dxs[i] = v * self.dt
            self.pos[i+1] = self.pos[i] + self.dxs[i]
            v = vnew
        self.finalV = v

    def solveMany(self, times = 1):
        """Do the Brownian motion for n times and return the averages of values
        """
        finalV = numpy.zeros(times)
        finalX = numpy.zeros(times)
        for i in range(times):
            self.reset()
            self.solve()
            finalV[i] = self.finalV
            finalX[i] = self.pos[-1]
        return finalX, finalV

def getRMSDisplacement(pos):
    """get root mean squared displacements of a particle """
    return numpy.mean(numpy.square(pos))

def solve_problem1():
    import pylab
    timesteps = [0.1, 0.01, 0.001]
    initV = [0, 10]
    # This is a dictionary with keys as (timestep,initV) and final velocity as
    # values.
    finalvecolicies = defaultdict(list)
    total = 5
    for dt in timesteps:
        for v in initV:
            print("Solving for %s timestep and %s initial velocty" % (dt,v))
            print("++ Generating %s samples" % total)
            pylab.figure()
            for i in range(total):
                a = Brownian(dt, v)
                a.solve()
                finalvecolicies[(dt,v)].append(a.finalV)
                pylab.plot(a.time, a.pos)
            pylab.xlabel("Time: dt is %s" % dt)
            pylab.ylabel("Position when init velocity is %s" % v)
            print("++ Saving plot")
            pylab.savefig("plot_{}dt_{}initv_{}.png".format(int(dt*1000), v, total))

    with open("results_{}.txt".format(total), "w") as f:
        f.write("samples,dt,initv,mean,variance,rms\n")
        for dt, v in finalvecolicies:
            vs = finalvecolicies[(dt, v)]
            f.write("%s,%s,%s,%s,%s,%s\n"%(
                total
                ,dt
                ,v
                ,numpy.mean(vs)
                ,numpy.std(vs)
                ,numpy.sqrt(numpy.mean(numpy.square(vs))))
                )
        print("Problem 1 is solevd")

def solve_problem2():
    import pylab
    print("Solving problem 2")
    rmsXList = []
    simTimeList = []
    displacements = []
    total = 100
    repeat = 20
    for i in range(total):
        runtime = i+1
        b = Brownian(0.01, 0, runtime = runtime)
        finalX, finalV = b.solveMany(repeat)
        f = numpy.sqrt(numpy.mean(numpy.square(finalX)))
        displacements.append(f)
    pylab.plot(displacements)
    pylab.savefig("{}x{}.png".format(total, repeat))
    pylab.show()


if __name__ == '__main__':
```

```
if len(sys.argv) < 2:
    print("USAGE: {} 1|2".format(sys.argv[0]))
    sys.exit()
if "1" == sys.argv[1]:
    solve_problem1()
elif "2" == sys.argv[1]:
    solve_problem2()
else:
    print("USAGE: {} 1|2".format(sys.argv[0]))
    sys.exit()
```