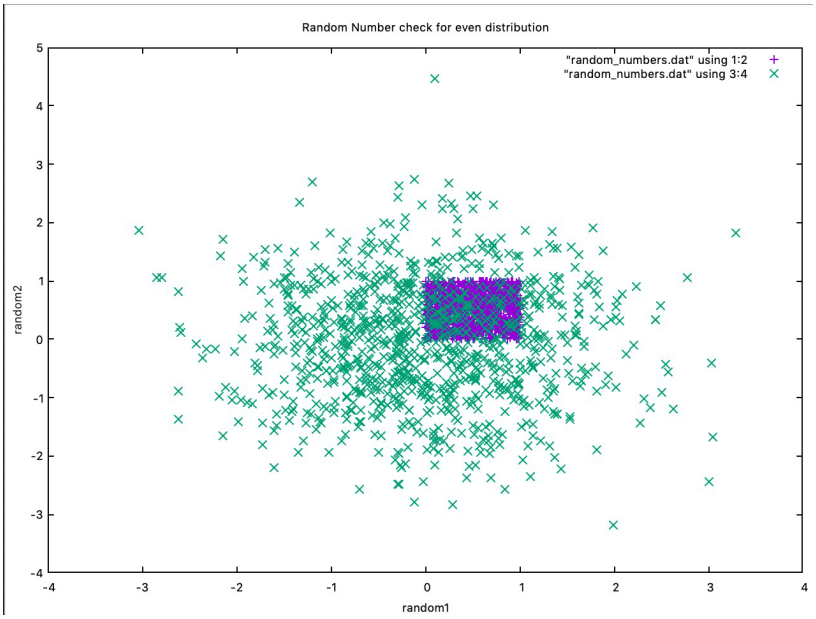
Act 12

Random # Generation

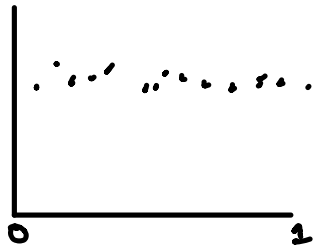
1. Constructor: Seed
# of random #'s

Destructor:

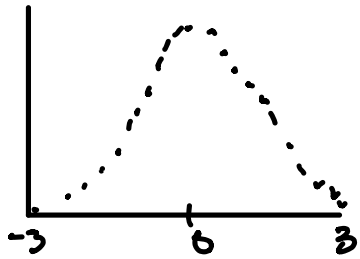
2. I plotted (uniform1, uniform2) w/ 1000 points.
(gaussian1, gaussian2)



3. uniform plots



Gaussian Plots



they do look random, and evenly distributed in the expected way.

4. 1,000,000 does reproduce the gaussian distribution best. As the point number decreases, the gaussian distribution gets worse.

Random Walking

1. $\langle \Delta x \rangle = \langle \Delta y \rangle = \pm \sqrt{2}$

$$\langle \Delta x^2 + \Delta x^2 \rangle = \langle r^2 \rangle = \frac{R^2}{N}$$

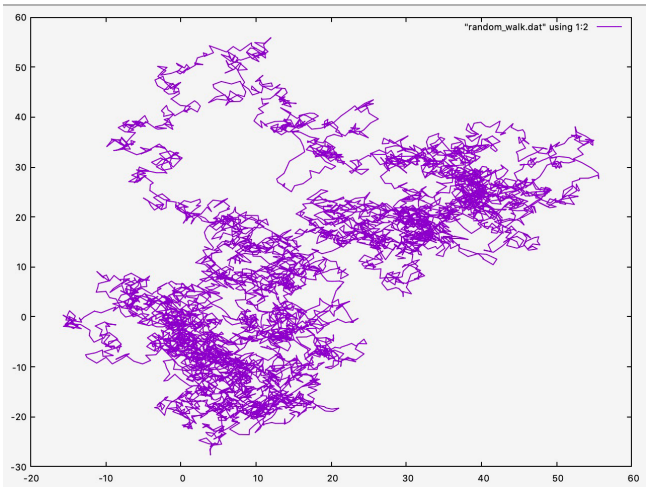
$$\langle 2\Delta x^2 \rangle = \frac{N r_{rms}^2}{N} = \quad 2\sqrt{2}^2 = r_{rms}^2$$

$$2 \cdot 2 = r_{rms}^2$$

$$r_{rms} = 2$$

2. ✓

3.



4. $R \approx \sqrt{N} \cdot r_{rms}$

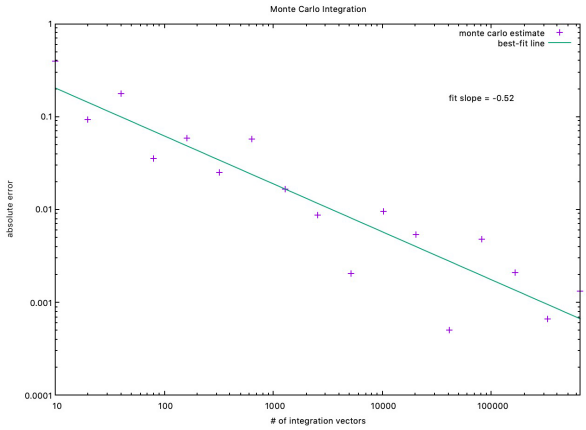
$R \approx 2\sqrt{N}$

$R \propto \sqrt{N}$

Monte Carlo Integration: Uniform and Gaussian Sampling

1. I would choose values 4 to 5 times the size of the σ value.

2.



1st plot

All of my plots had a fit slope in between -.5 and -.6. The y intercept was slightly variable (between .1 and 1) and the data points were always very different.

3.

	5	8	10	
Slope 1	-.44	-.34	-.16	
Slope 2	-.55	-.21	-.09	
Slope 3	-.50	-.21	-.06	→ not linear at all

Monte Carlo seems to work well for 4 and 5 D, but eventually stops working when D becomes too large.

4. Should return $(x\_sum)^2$

~ Yes D=100 did work.

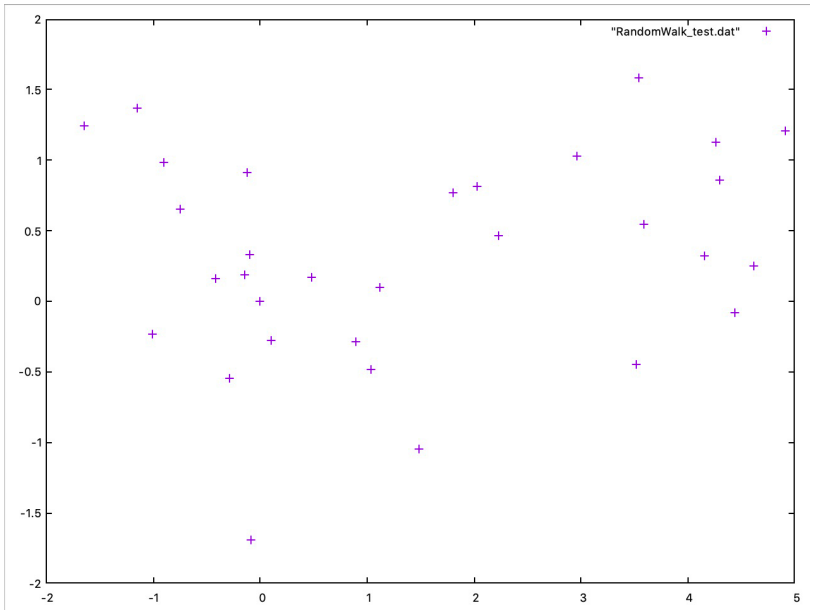
Monte Carlo Integration: GSL Routines

1. ✓

2. vegas always had the lowest sigma value

C++ Class for a Random Walk

1.



2. The new code has many of the details in a seperate file, but the old code is more clear to read. The old code would be easier to modify, but the new one would take a bit more work if any changes needed to be made.

3. Extend to 3 dimensions by adding a z direction.

   maybe have unequal step sizes for x and y directions.