Untitled

April 2, 2023

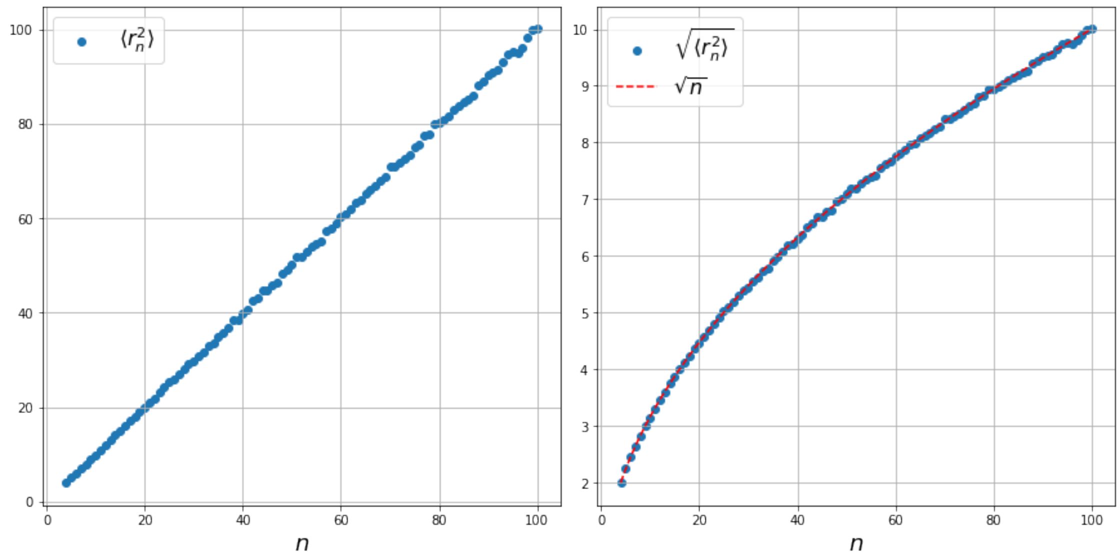
1 6.1.a

[91]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[174]:
```
n1 = 100
w1 = 2e4

np.random.seed(0)
r2ave1 = []

for step in range(4, n1 + 1):
    r2 = [0]
    for walk in range(int(w)):
        pos = {'x':0, 'y':0}
        for _ in range(step):
            s = np.random.randint(0, 4)
            if s == 0:
                pos['x'] += 1
            elif s == 1:
                pos['x'] -= 1
            elif s == 2:
                pos['y'] += 1
            else:
                pos['y'] -= 1
        r2.append((pos['x']**2 + pos['y']**2))
    r2ave1.append(np.mean(r2))
```

[210]:
```
fig, ax = plt.subplots(1, 2, constrained_layout = True, figsize = (12, 6))

ax[0].scatter(np.arange(4, n1 + 1), r2ave1, label = r'$\langle r_n^2 \rangle$')
ax[0].set_xlabel(r'$n$', fontsize = 18)
ax[0].legend(fontsize = 16)
ax[0].grid()

ax[1].scatter(np.arange(4, n1 + 1), np.sqrt(r2ave1), label = r'$\sqrt{\langle␣
↪r_n^2 \rangle}$')
```

```
ax[1].plot(np.arange(4, n1 + 1), np.sqrt(np.arange(4, n1 + 1)), ls = '--',
 ↪color = 'red', label = r'$\sqrt{n}$')
ax[1].set_xlabel(r'$n$', fontsize = 18)
ax[1].legend(fontsize = 16)
ax[1].grid()

plt.show()
```



Through eyeball fit, we observe that the Flory exponent $\nu \approx 1/2$

---

2 6.1.b

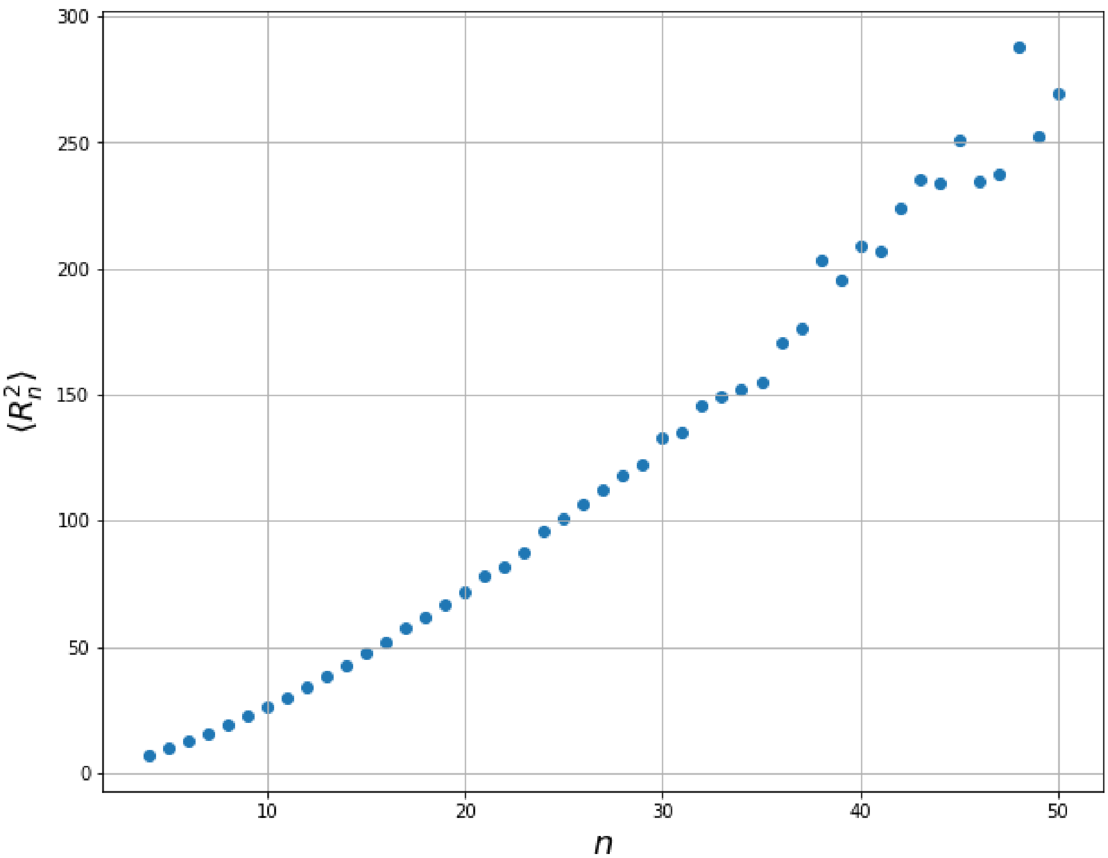
[34]:
```
n = 50
w = 2e4

#origin
start = n // 2
```

[38]:
```
r2ave = []
choice = {'right', 'left', 'up', 'down'}
opp = {
    'right':'left',
    'left':'right',
    'up':'down',
    'down':'up'
```

```
}

np.random.seed(0)
for step in range(4, n+1):
    r2 = [0]
    for walk in range(int(w)):
        # since n = 50, we use 100x100 max array
        a = np.full((2*n+1, 2*n+1), False)
        x = y = start
        dead = False
        moved = False
        prev = None
        for _ in range(step):
            a[x, y] = True

            if not moved:
                s = np.random.choice(list(choice))
                if s == 'right':
                    x += 1
                elif s == 'left':
                    x -= 1
                elif s == 'up':
                    y += 1
                elif s == 'down':
                    y -= 1

                moved = True
                prev = opp[s]
            else:
                avail = choice.difference({prev})
                s = np.random.choice(list(avail))
                if s == 'right' and not a[x+1, y]:
                    x += 1
                    prev = opp[s]
                elif s == 'left' and not a[x-1,y]:
                    x -= 1
                    prev = opp[s]
                elif s == 'up' and not a[x, y+1]:
                    y += 1
                    prev = opp[s]
                elif s == 'down' and not a[x, y-1]:
                    y -= 1
                    prev = opp[s]
                else:
                    dead = True
                    walk -= 1
                    break
```

```
        if not dead and len(r2) < w:
            r2.append((x - start)**2 + (y-start)**2)
    r2ave.append(np.mean(r2))
```

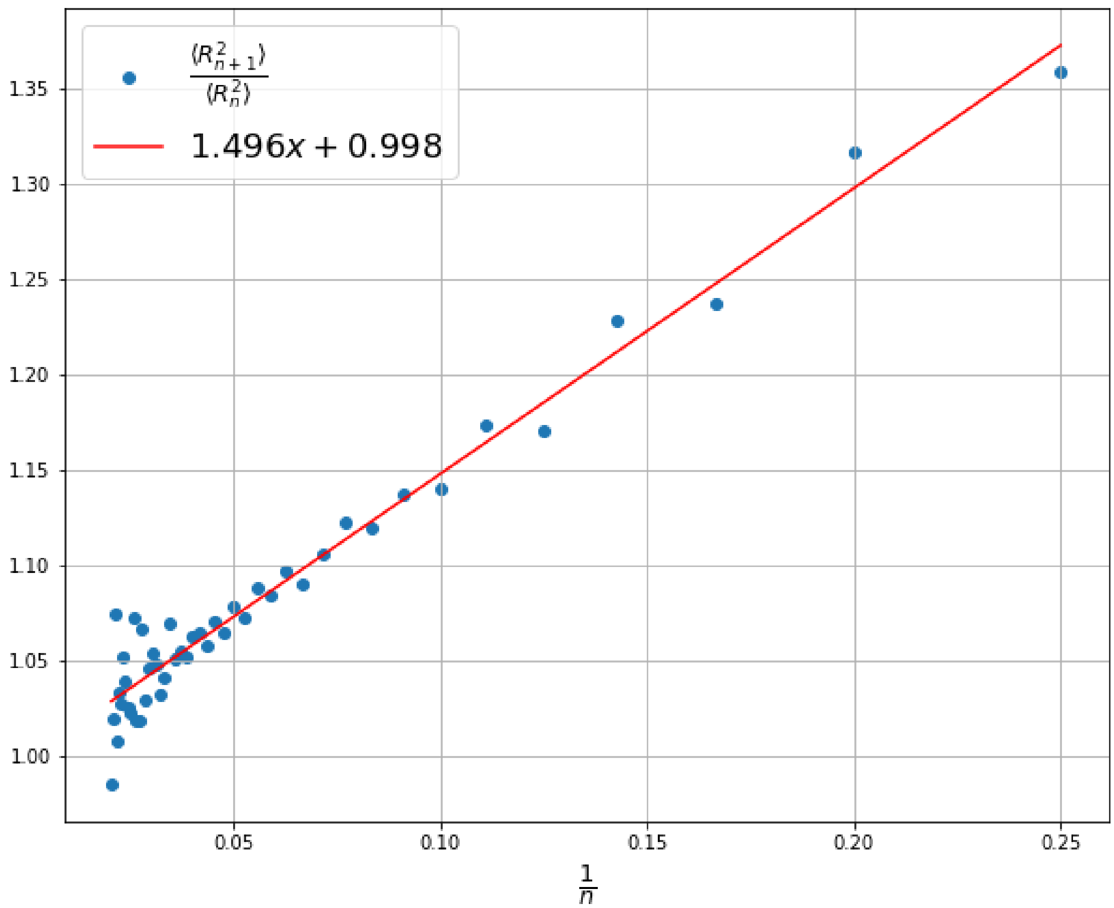
[76]:
```
plt.figure(figsize = (10, 8))
plt.scatter(np.arange(4, n+1), r2ave)
plt.xlabel(r'$n$', fontsize = 18)
plt.ylabel(r'$\langle R_n^2 \rangle$', fontsize = 18)
plt.grid()
plt.show()
```



[43]:
```
arr1 = r2ave[1:47]
arr2 = r2ave[0:46]

arr1 = np.array(arr1)
arr2 = np.array(arr2)

arr3 = arr1 / arr2
```

[67]:
```
ns = []

for i in range(4, n):
    ns.append(1 / i)
```

[84]:
```
plt.figure(figsize = (10, 8))
plt.scatter(ns, (arr3), label = r'$\frac{\langle R_{n+1}^2 \rangle}{\langle␣
↪R_n^2 \rangle}$')
plt.plot(ns, 1.49654069*np.array(ns) + 0.99873934, color = 'red', label = r'$1.
↪496x + 0.998$')
plt.xlabel(r'$\frac{1}{n}$', fontsize = 18)
plt.legend(fontsize = 18)
plt.grid()
plt.show()
```



Since

$$\frac{\langle R_{n+1}^2 \rangle}{\langle R_n^2 \rangle} = \frac{(n+1)^{2\nu}}{n^{2\nu}} = \left(\frac{n+1}{n}\right)^{2\nu} = \left(1 + \frac{1}{n}\right)^{2\nu}$$

Because n is large then $1/n \ll 1$, and so by first order Taylor expansion, we have that

$$\boxed{\frac{\langle R_{n+1}^2 \rangle}{\langle R_n^2 \rangle} \approx 1 + 2\nu \frac{1}{n}}$$

So since our our slope is 2ν, if follows that $\nu \approx 3/4$.

If we were to use this using $\langle r_n^2 \rangle$ from (a), then

[153]:
```
arr1_1 = r2ave1[1:len(r2ave1)]
arr2_1 = r2ave1[0:len(r2ave1) - 1]
arr3_1 = np.array(arr1_1) / np.array(arr2_1)
ns1 = []
for i in range(4, n1):
    ns1.append(1 / i)
np.polyfit(ns1, arr3_1, deg = 1)
```

[153]:
```
array([1.0046373, 0.9999193])
```

As expected, $2\nu \approx 1$ and thus $\nu \approx 0.5$ for (a) which is the same using the square root approach.

---

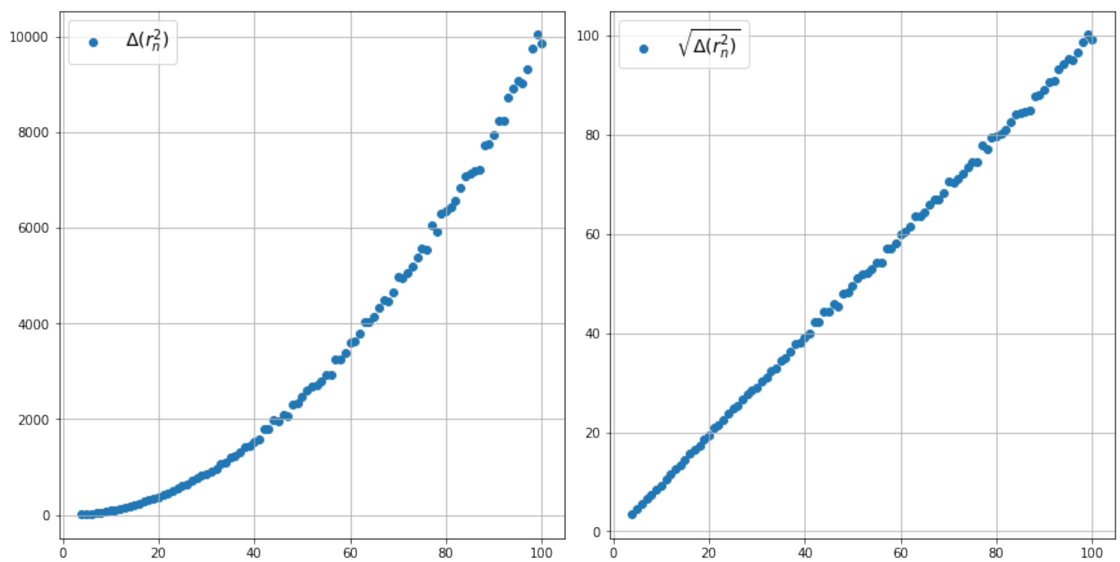
3 6.1.c

[157]:
```
n1 = 100
w1 = 2e4

np.random.seed(0)
r2ave2 = []
r4ave = []

for step in range(4, n1 + 1):
    r2 = [0]
    r4 = [0]
    for walk in range(int(w)):
        pos = {'x':0, 'y':0}
        for _ in range(step):
            s = np.random.randint(0, 4)
            if s == 0:
                pos['x'] += 1
```

```
            elif s == 1:
                pos['x'] -= 1
            elif s == 2:
                pos['y'] += 1
            else:
                pos['y'] -= 1
        r2.append((pos['x']**2 + pos['y']**2))
        r4.append(r2[-1]**2)
    r2ave2.append(np.mean(r2)**2)
    r4ave.append(np.mean(r4))
```

[158]:
```
dr2 = np.array(r4ave) - np.array(r2ave2)
```

[219]:
```
fig, ax = plt.subplots(1, 2, constrained_layout = True, figsize = (12, 6))

ax[0].scatter(np.arange(4, n1 + 1), dr2, label = r'$\Delta(r_n^2)}$')
ax[0].legend(fontsize = 14)
ax[0].grid()

ax[1].scatter(np.arange(4, n1 + 1), np.sqrt(dr2), label =␣
 ↪r'$\sqrt{\Delta(r_{n}^2)}$')
ax[1].legend(fontsize = 14)
ax[1].grid()

plt.show()
```



Thus we see that $x \approx 1$

[188]:

```
xp = np.arange(4, n1 + 1)

plt.figure(figsize = (10, 8))
plt.scatter(xp, r2ave1, label = r'$\langle R_n^2 \rangle$')
plt.plot(xp, np.array(r2ave1) + xp, 'g--')
plt.plot(xp, np.array(r2ave1) - xp, 'g--')
plt.fill_between(xp, np.array(r2ave1) + xp, np.array(r2ave1) - xp, facecolor =␣
↪'gray', alpha = .15)
plt.xlabel(r'$n$', fontsize = 18)
plt.legend(fontsize = 18)
plt.show()
```