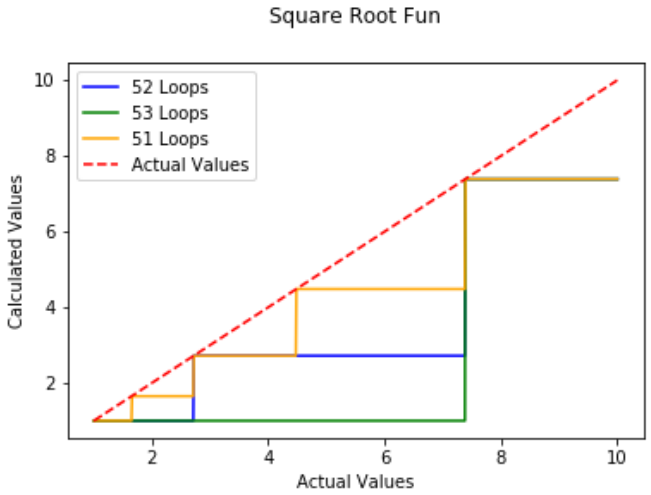
In [4]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import colors
import math
import seaborn as sns

np.set_printoptions(threshold=np.nan)

N=1001 # number of floats
original_array = np.zeros(N)
array = np.zeros(N) #Creating array of N zeros
new_array = array
start = float(1.0) #Starting value of array
dt = float(0.009) #time step
loops = 52 #number of loops for square rooting and squaring

#generate initial array
for i in range (0, N):
    array[i] = start + i*dt
    original_array[i] = start + i*dt

for i in range(0, loops):
    array[:] = [np.sqrt(x) for x in array]


for i in range(0, loops):
    array[:] = [x**2 for x in array]

print('52 Loops: Accurate Values')

for i in range(0, N):
    if abs(new_array[i] - original_array[i]) < 0.005:
        print(new_array[i])


array2 = np.zeros(N) #Creating array of N zeros
new_array2 = array2
start = float(1.0) #Starting value of array
dt = float(0.009) #time step
loops2 = 53 #number of loops for square rooting and squaring

#generate initial array
```

```
52 Loops: Accurate Values
1.0
2.718281808182473
7.389055988695776
53 Loops: Accurate Values
1.0
7.389055988695776
51 Loops: Accurate Values
1.0
2.718281808182473
4.481689053641878
7.389055988695776
```



The values that remain accurate are all powers of e. For 51 loops, e^even number is accurate, for 52 loops, e^n is accurate, and for 53 loops, e^n/2 is accurate. I am not sure why e remains accurate when other values do not.

In [ ]: