3d)
The dominant eigenvalue is 1.0000000000000002 which I am assuming is meant to be 1 but is slightly over due to floating point error in some calculations. The respective eigenvector is:

```
[0.0102113  0.01022957 0.01004233 0.01075901 0.00972892 0.00951361
 0.01025973 0.01011713 0.01095411 0.01016109 0.0096846  0.01073044
 0.00926431 0.00922182 0.0104885  0.00994969 0.00871076 0.01000429
 0.0096862  0.00951964 0.01018877 0.0094845  0.01053894 0.01062212
 0.01034435 0.00934375 0.0100773  0.00987915 0.01041015 0.00999376
 0.01036215 0.01066164 0.00999422 0.00873251 0.01007994 0.0096642
 0.0096195  0.01015154 0.01052289 0.01049354 0.01007868 0.01029074
 0.01086413 0.00995877 0.00977635 0.00926963 0.00988486 0.01071303
 0.00972796 0.00987513 0.00959909 0.00874806 0.00978713 0.00852708
 0.01042781 0.01078748 0.01030365 0.01008664 0.01010964 0.00999012
 0.00980689 0.00963091 0.01063079 0.0099368  0.00969259 0.01020189
 0.00925353 0.00941178 0.01006868 0.01034721 0.01052003 0.00973715
 0.01047055 0.00946661 0.00952008 0.0100603  0.01013914 0.00900002
 0.01066497 0.01032412 0.01083284 0.00961408 0.00976624 0.01023393
 0.00953726 0.00979244 0.0103889  0.0103317  0.01068876 0.01039886
 0.0111182  0.00947061 0.00882996 0.00936265 0.01001645 0.0107688
 0.01073303 0.00992018 0.01049022 0.00961288]
```

3e)
We can represent the Internet and website links as an nxn matrix, where each index i,j is the probability of going from website i to website j. This is a stochastic matrix, and we can find the dominant eigenvector which indicates if you were to keep ‘surfing the web’ the probability of you ending up on a given page. This is because for any k number of iterations, $x^{(k)} = A^k x^{(0)}$ and we can think of each A as each time one surfs the web and they have the ability to jump pages.

Code Question 1)

```
import numpy as np
import math

f = lambda x : np.sqrt(1-(5/9)*(np.sin(x)**2)) #function

#Trapezoid method with k^2 = 5/9 as b=2, a=3
a = 0
b = math.pi/2
size = 10**6
x = np.linspace(a, b, size)

h = (b-a)/size #interval size
y = f(x)

int = 0 #Does numerical integration following the trapezoid method
for i in range(1, size):
    int += 2*y[i]
int = 12*((int + y[0]+y[99])*h/2)
print(int)

h = 1/25 #All other methods of ellipse perimeter
print(2*math.pi*np.sqrt((2**2+3**2)/2), 'method 1')
print(math.pi*(3*(5)-np.sqrt((3*3+2)*(3+3*2))), 'ramanujan 1')
print(math.pi*5*(1+(3*h)/(10+np.sqrt(4+3*h))), 'ramanujan 2')
# I pulled the approximations from a Matt Parker video:
# https://www.youtube.com/watch?v=5nW3nJhBHL0&t=1058s
```

Code Question 3)

```
import numpy as np
import csv

M = np.zeros((100, 100))

col = 0
count = 0
with open('stochastic_matrix.csv') as csv_file: #Reads in file
    csv_reader = csv.reader(csv_file, delimiter=',')
    for row in csv_reader:
        for i in range(0, len(row)):
            M[col, i] = float(row[i])
        col += 1

x = np.random.uniform(0, 5, 100) #Generates random vector
```

```
def power(M, x): #Does power iteration
    x_k = x
    for i in range(0, 100):
        x_k1 = np.matmul(M, x_k)
        x_k = x_k1/np.linalg.norm(x_k1)
    return x_k

#res is eigenvector and quot is eigenvalue
res = power(M, x)
res = res/np.sum(res)
quot = np.matmul(np.matmul(res.T, M), res)/(np.matmul(res.T, res))
print(res, np.sum(res), quot)
```