1.1)

The problem is to use numerical integration to find the perimeter of an ellipse. To perform numerical integration, I use the trapezoid method. The merits of the method are that for a given function, it’s very easy to calculate as it involves just evaluating the function and taking a sum. Additionally, the method is scalable to any function over any domain. The trapezoid method does converge to the true result as the number of points you use goes to infinity. Furthermore, as the formula for the trapezoid method is just taking a sum, numerical integration should be backwards stable. One restriction on numerical integration is that the function you are integrating will have to be continuous on the domain you are integrating it over. Additionally, as you are taking a sum of functions, the function will need to be well conditioned to guarantee good results.

1.2)

The problem is to perform numerical integration using Gaussian quadrature, which uses a system of orthogonal polynomials to pick points to calculate a weighted sum that approximates the integral. There are issues with simplicity, as to perform Gaussian quadrature you have to know the weights, which requires you to differentiate and evaluate a polynomial, which gets harder for higher degrees. However, as it can be seen in the problem, at least for this equation, the integration converges quickly. Additionally, though the form of quadrature we discussed was over [-1,1], you can scale it to any interval. Like 1.1, Gaussian quadrature is also backwards stable as you are taking a sum. Quadrature has the same issue with the function being continuous, and as you are using orthogonal polynomials, you could still run into the issue with ill conditioning.

3)

The problem is to find the dominant eigenvector and eigenvalue of a column stochastic matrix. To calculate this, I use power iteration. The merit of the method is that it will globally converge from any starting point to the dominant eigenvector. Additionally, it is more efficient at finding the eigenvector than other methods. Power iteration is scalable to any matrix of any size, and it’s not necessarily backwards stable, as you are performing matrix vector multiplication. In the case of this problem, where the matrix is column stochastic, it will converge to the dominant eigenvector. However, generally, convergence cannot be guaranteed. A practical limitation of power iteration is that though it can find the dominant eigenvector, it becomes increasingly difficult to find all other eigenvectors and eigenvalues.

Math 125 Proj 2

1a) $L=\int_{-\pi}^{\pi}\sqrt{(-a\sin t)^2+(b\cos t)^2}\,dt=\int_{-\pi}^{\pi}a\sqrt{\cos^2 t+\sin^2 t}\,dt=2\pi a$

b) $L=\int_{-\pi}^{\pi}\sqrt{(-a\sin t)^2+(b\cos t)^2}\,dt=\int_{-\pi}^{\pi}\sqrt{a^2\sin^2 t+b^2\cos^2 t}\,dt$

$=\int_{-\pi}^{\pi}\sqrt{a^2(\sin^2 t+\frac{b^2}{a^2}\cos^2 t)}\,dt=a\int_{-\pi}^{\pi}\sqrt{\sin^2 t+(b^2/a^2)(1-\sin^2 t)}\,dt$

$=a\int_{-\pi}^{\pi}\sqrt{\frac{b^2}{a^2}+\sin^2 t-\frac{b^2}{a^2}\sin^2 t}\,dt=a\int_{-\pi}^{\pi}\sqrt{\frac{b^2}{a^2}+(1-\frac{b^2}{a^2})\sin^2 t}\,dt$

$1-\frac{b^2}{a^2}=k^2$

$=a\int_{-\pi}^{\pi}\sqrt{(1-k^2)+k^2\sin^2 t}\,dt=a\int_{-\pi}^{\pi}\sqrt{1+k^2(1-\sin^2 t)}\,dt$

$=a\int_{-\pi}^{\pi}\sqrt{1-k^2\cos^2 t}\,dt=a\int_{-\pi}^{\pi}\sqrt{1-k^2\sin^2 t}\,dt$ as both are positive only and $-\pi$ to π is one rotation so

$=2a\int_{0}^{\pi}\sqrt{1-k^2\sin^2 t}\,dt=4a\int_{0}^{\pi/2}\sqrt{1-k^2\sin^2 t}\,dt$ as splits cycle. we can do this again as positive

c)

$12\int_{0}^{\pi/2}\sqrt{1-\frac{5}{9}\sin^2 t}\,dt$

Use trapezoid method, I use $n=10^6$ to get error $\le 10^{-6}$

error analysis

So from class error is:

$e_n\le\frac{\max_{x\in[a,b]}|f''(x)|\,(b-a)^3}{12n^2}$

$e_n\le\frac{|5/6|(\frac{\pi}{2})^3}{12\cdot(10^6)^2}=\frac{\frac{5\pi^3}{48}}{12\cdot 10^{12}}\le 10^{-6}$

$f(x)=\sqrt{1-\frac{5}{9}\sin^2 x}$

$f''(x)=\frac{-5(5\sin^4 x-9\sin^2 x+9\cos^2 x)}{3(9-5\sin^2(x))^{3/2}}$

max on $[0,\frac{\pi}{2}]=5/6$

1 c) con. Using my code I get
perimeter = 15.86543 94318 94304. (code at end)

1 d) Alternative perimeter approximations
$2\pi\sqrt{\frac{a^2+b^2}{2}}$: 16.01904...

$\begin{cases} \pi\left(3(a+b)-\sqrt{(3a+b)(a+3b)}\right) : 15.86543 \\ \pi(a+b)\left(1+\frac{3h}{10+\sqrt{4-3h}}\right) : 15.864654 \end{cases}$

$h = \frac{(a-b)^2}{(a+b)^2}$

Ramanujan approximations, first one is from Matt Parker video, link is in code. Both Ramanujan approxs. are within .001 of the numerical integration.

1.2 a) You can use Gaussian quadrature to get weights and a polynomial that is orthogonal to do the integral.

b) For n=2 $\int_{-1}^{1} f(x)\,dx \approx \sum w_i f(x_i) = f(-\frac{1}{\sqrt{3}}) + f(\frac{1}{\sqrt{3}})$
As $w_i = 1$

So $\int_{-1}^{1} f(x)\,dx \approx 1.693$

For n=3 $\int_{-1}^{1} f(x)\,dx \approx \sum w_i f(x_i) = \frac{5}{9} f(-\sqrt{\frac{3}{5}}) + \frac{8}{9} f(0) + \frac{5}{9} f(\sqrt{\frac{3}{5}})$

$\int_{-1}^{1} f(x)\,dx \approx 1.712$

For n=4
Weights from wikipedia and roots from problem
$\int_{-1}^{1} f(x)\,dx \approx \sum w_i f(x_i) = 1.7112$ rounded since I'm writing this out.

1.2 c) Normal PDF is $U = \frac{1}{\sqrt{2\pi}} e^{-x^2}$ so we can take

$$\int_{-1}^{1} \frac{1}{\sqrt{2\pi}} e^{-x^2} dx = \frac{1}{\sqrt{2\pi}} \int_{-1}^{1} e^{-x^2} dx$$

Using a numerical solver $\int_{-1}^{1} \frac{1}{\sqrt{2\pi}} e^{-x^2} dx \approx 0.683$.

In comparison to results from b, take answers and multiply by $1/\sqrt{2\pi}$.

$n=2$: $1.693/\sqrt{2\pi} \approx 0.675$, $e_2 = 0.0017$
$n=3$: $1.712/\sqrt{2\pi} \approx 0.792$ $e_3 = 0.00003$
$n=4$: $1.7111/\sqrt{2\pi} \approx 0.683$ $e_4 = 0.000001$.
] same rounding done

3 a) Note that A and A^T have the same eigenvalues. As $A^T v = \lambda v \rightarrow \det(A^T v - \lambda I v)$
$\rightarrow \det((Av - \lambda I v)^T) = \det(Av - \lambda I v)$

A^T is row stochastic and let $v = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$

$Av = \lambda v$ $A\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$ $\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \\ a_{n1} & & a_{nn} \end{bmatrix}\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$

The sum of each row is 1, and get

$\begin{bmatrix} \sum a_{1i} \\ \vdots \\ \sum^{n} a_{in} \end{bmatrix} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$ So $\lambda = 1$ is an eigenvalue of A.

b) Using the circle theorem on A^T, all eigenvalues fall within the circle centered at a_{ii} w/ radius $\sum_{i \neq n} a_{in}$ as $a_{ii} + \sum_{i \neq n} a_{in} = 1$, all eigenvalues will be ≤ 1, and as $\lambda = 1$ is an eigenvalue, it is the largest one

3c) $x^* = Ax^*$. As $\lambda = 1$, then we can say x^* is an eigenvector of A with eigenvalue of 1 as $Ax^* = 1 \cdot x^*$ when it has converged, and as it is the vector associated with the largest eigenvalue, it is the dominant eigenvector.

3d and e are on next page.

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)
```