Q1 Part 1

For Matrix A1 the Rayleigh Power Method converges in roughly half the iterations of the Scaling Power Method; the eigenvalues are not close to one another; this drastically improves the convergence.

Q1 Part 1

A1	**Scaling Power**				
k	**Approx. Eigenvalue Lambda 1**	**Approx. Error**	**Numerical Rate of Convergence**	**Theoretical Rate of Convergence**	**Normalized Difference**
1	10.00000000	2.12701572		0.12701664	
2	8.10000000	0.22701572	0.10672969	0.01613323	0.190000
3	7.90123457	0.02825029	0.12444200	0.00204919	0.024539
4	7.87656250	0.00357822	0.12666144	0.00026028	0.003123
5	7.87343781	0.00045353	0.12674799	0.00003306	0.000397
6	7.87304107	0.00005679	0.12521888	0.00000420	0.000050
7	7.87299068	0.00000640	0.11270107	0.00000053	0.000006
8	7.87298428	0.00000000	0.00000000	0.00000007	0.000001
9					
10					
Final lambda 1	**7.872984277**				
lambda 2	1				

Q1 Part 1

A1	**Rayleigh Power**				
k	**Approx. Eigenvalue Lambda 1**	**Approx. Error**	**Numerical Rate of Convergence**	**Theoretical Rate of Convergence**	**Normalized Difference**
1	7.84137931	-0.03160403		0.12701665	
2	7.87247122	-0.00051212	0.01620441	0.01613323	0.003965107
3	7.87297508	-0.00000826	0.01613025	0.00204919	0.000064003
4	7.87298321	-0.00000013	0.01587710	0.00026028	0.000001033
5	7.87298334	0.00000000	0.00000000	0.00003306	0.000000017
6					
7					
8					
9					
10					
Final lambda 1	**7.872983344**				
lambda 2	1				

For Matrix A2 the eigenvalues are very close. The scaling method converges before the rayleigh method.

Q1 Part 1

A2	**Scaling Power**				
k	**Approx. Eigenvalue Lambda 1**	**Approx. Error**	**Numerical Rate of Convergence**	**Theoretical Rate of Convergence**	**Normalized Difference**
1	8.00000000	0.12701666		0.97734270	
2	7.25000000	-0.62298334	0.10471506	0.95519875	0.093750
3	7.17241379	-0.70056955	0.11554568	0.93355653	0.010702
4	7.16346154	-0.70952181	0.11677852	0.91240466	0.001248
5	7.16241611	-0.71056724	0.11677852	0.89173203	0.000146
6	7.16229385	-0.71068949	0.11554568	0.87152779	0.000017
7	7.16227955	-0.71070379	0.10471506	0.85178132	0.000002
8	7.16227788	-0.71070546	0.00000000	0.83248226	0.000000
9					
10					
final lambda 1	**7.162277882**				
lambda 2	7				

Q1 Part 1

A2	**Rayleigh Power**				
k	**Approx. Eigenvalue Lambda 1**	**Approx. Error**	**Numerical Rate of Convergence**	**Theoretical Rate of Convergence**	**Normalized Difference**
1	7.10738255	-0.05474615		0.97736306	
2	7.11050604	-0.05162266	0.94294589	0.95523854	0.000439472
3	7.11212881	-0.04999989	0.96856477	0.93361486	0.000228221
4	7.11370331	-0.04842539	0.96850999	0.91248068	0.000221382
5	7.11524883	-0.04687987	0.96808446	0.89182490	0.000217260
6	7.11676486	-0.04536384	0.96766144	0.87163671	0.000213067
7	7.11825069	-0.04387802	0.96724646	0.85190552	0.000208778
8	7.11970569	-0.04242301	0.96683978	0.83262098	0.000204405
9	7.12112934	-0.04099936	0.96644151	0.81377299	0.000199959
10	7.1225212	-0.03960750	0.96605178	0.00000000	0.000195454
final lambda 1	**7.162128704**				
lambda 2	7				

For Matrix A3 the eigenvalues are very close. The scaling method converges before the rayleigh method. The dominant eigenvalue is not unique. <u>Both methods are very slow to converge</u>.

Q1 Part 1

A3	**Scaling Power**				
k	**Approx. Eigenvalue Lambda**	**Approx. Error**	**Numerical Rate of Convergence**	**Theoretical Rate of Convergence**	**Normalized Difference**
1	4.00000000	0.99700000		0.99900100	
2	3.75000000	0.74700000	0.74924774	0.99800300	0.062500
3	3.60000000	0.59700000	0.79919679	0.99700599	0.040000
4	3.50000000	0.49700000	0.83249581	0.99600998	0.027778
5	3.42857143	0.42557143	0.85628054	0.99501497	0.020408
6	3.37500000	0.37200000	0.87411883	0.99402094	0.015625
7	3.33333333	0.33033333	0.88799283	0.99302792	0.012346
8	3.30000000	0.29700000	0.89909183	0.99203588	0.010000
9	3.27272727	0.26972727	0.90817264	0.99104484	0.008264
10	3.25	0.24700000	0.91573980	0.99005478	0.006944
final lambda 1	**3.0030000000**				
lambda 2	3				

Q1 Part 1

A3	**Rayleigh Power**				
k	**Approx. Eigenvalue Lambda**	**Approx. Error**	**Numerical Rate of Convergence**	**Theoretical Rate of Convergence**	**Normalized Difference**
1	3.27586207	0.27286210		0.99900101	
2	3.36956522	0.36656524	1.34340845	0.99800301	0.028604119
3	3.37584255	0.37284257	1.01712472	0.99700602	0.001862949
4	3.35387183	0.35087186	0.94107241	0.99601002	0.006508216
5	3.32593228	0.32293231	0.92037108	0.99501501	0.008330535
6	3.29899882	0.29599884	0.91659717	0.99402100	0.008098020
7	3.27486878	0.27186880	0.91847927	0.99302798	0.007314353
8	3.25371401	0.25071404	0.92218760	0.99203595	0.006459728
9	3.23524495	0.23224498	0.92633416	0.99104492	0.005676301
10	3.219082587	0.21608261	0.93040812	0.99005487	0.004995716
final lambda 1	**3.002999973**				
lambda 2	3				

<u>Q1 Part 2</u>

The new first guess column vector $X^{(0)}$ given in part 2 is perpendicular (<u>orthogonal</u>) to the eigenvector associated with the dominant eigenvalue for Matrix A1 from part 1.

For the Power Method, a required condition for the choice of the initial first guess column vector $X^{(0)}$ is that it must have a non-zero component in the direction of the dominant eigenvector. This statement means that the initial first guess column vector $X^{(0)}$ **should not be perpendicular(orthogonal) to the dominant eigenvector X_1.**

When an exactly perpendicular first guess column vector $X^{(0)}$ is used, the Power Method will converge to the eigenvector corresponding to the second largest eigenvalue. This is shown on the attached output "Q1 Part 2 Scaling Power Exact" and "Q1 Part 2 Rayleigh Power Exact" generated using a very exact input of the perpendicular first guess column vector $X^{(0)}$.

When an approximation of the perpendicular first guess column vector $X^{(0)}$ is used, the Power Method will converge to the eigenvector associated with the dominant eigenvalue. This is shown on the attached output "Q1 Part 2 Scaling Power Approximate " and "Q1 Part 2 Rayleigh Power Approximate" generated by using an approximate input of the perpendicular first guess column vector $X^{(0)}$.

The approximation of $\mathbf{X^{(0)}}$ meets the condition of having **at least a <u>very tiny</u> non-zero component in the direction of the dominant eigen vector** and will therefore eventually converge to the eigenvector associated with the dominant eigenvalue.

In practice, round-off errors or an approximation of the perpendicular(orthogonal) vector usually prevent this problem.

Note:
From Linear Algebra we know that two vectors , X_1 and $X^{(0)}$, are perpendicular(orthogonal) if $(X_1)^T * (X^{(0)}) = 0$. Taking the dominant eigenvector from part 1 and the new initial first guess column vector $X^{(0)}$ from part 2, we find that $(X_1)^T * X^{(0)}$ is approximately equal to zero. Therefore the two vectors are approximately perpendicular(orthogonal). Using exact math, the two vectors should be exactly perpendicular(orthogonal).

<u>Results for Part 2 are shown on the next two pages.</u>

Q1 Part 2

A1	Scaling Power	"Exact"			
k	**Approx. Eigenvalue Lambda 1**	**Approx. Error**	**Numerical Rate of Convergence**	**Theoretical Rate of Convergence**	**Normalized Difference**
1	1.77459667	0.77459662		0.99999995	
2	1.11088342	0.11088337	0.14314983	0.99999990	0.374008
3	1.01267824	0.01267818	0.11433801	0.99999985	0.088403
4	1.00159019	0.00159013	0.12542290	0.99999979	0.010949
5	1.00020166	0.00020161	0.12678663	0.99999974	0.001386
6	1.00002561	0.00002556	0.12676733	0.99999969	0.000176
7	1.00000325	0.00000320	0.12524822	0.99999964	0.000022
8	1.00000041	0.00000036	0.11289507	0.99999959	0.000003
9	1.00000005				
10	0				
Final lambda 1	**1.000000052**				
lambda 2	1				

Q1 Part 2

A1	Rayleigh Power	"Exact"			
k	**Approx. Eigenvalue Lambda 1**	**Approx. Error**	**Numerical Rate of Convergence**	**Theoretical Rate of Convergence**	**Normalized Difference**
1	0.97570215	-0.02429784		1.00000000	
2	0.99959696	-0.00040304	0.01658739	1.00000000	0.024489857
3	0.99999349	-0.00000650	0.01613643	1.00000001	0.000396694
4	0.99999990	-0.00000010	0.01587720	1.00000001	0.000006400
5	1.00000000	0.00000000	0.00000000	1.00000001	0.000000103
6					
7					
8					
9					
10					
Final lambda 1	**0.999999998**				
lambda 2	1				

Q1 Part 2

A1	Scaling Power	"Approximate"			
k	**Approx. Eigenvalue Lambda 1**	**Approx. Error**	**Numerical Rate of Convergence**	**Theoretical Rate of Convergence**	**Normalized Difference**
1	1.77459600	0.77459600		0.12701666	
2	1.11088214	0.11088214	1.10883433	0.01613323	0.374008
3	1.01266887	0.01266888	1.01452408	0.00204919	0.088410
4	1.00151728	0.00151728	1.00162552	0.00026028	0.011012
5	0.99962838	-0.00037162	1.00027489	0.00003306	0.001886
6	0.99551042	-0.00448958	1.00059912	0.00000420	0.004119
7	0.96429481	-0.03570518	1.00453882	0.00000053	0.031356
8	2.00282079	1.00282079	0.84967827	0.00000007	1.076980
9	6.08437376	5.08437377	0.30469505	0.00000001	2.037902
10	7.57901563	6.57901563	0.16435536	0.00000000	0.245653

final lambda 1	**7.872983182**
lambda 2	1

Q1 Part 2

A1	Rayleigh Power	"Approximate"			
k	**Approx. Eigenvalue Lambda 1**	**Approx. Error**	**Numerical Rate of Convergence**	**Theoretical Rate of Convergence**	**Normalized Difference**
1	0.97570215	-6.89728119		0.12701665	
2	0.99959696	-6.87338638	0.99653562	0.01613323	0.024489864
3	0.99999351	-6.87298983	0.99994231	0.00204919	0.000396708
4	1.00000074	-6.87298260	0.99999895	0.00026028	0.000007235
5	1.00005256	-6.87293078	0.99999246	0.00003306	0.000051822
6	1.00325676	-6.86972658	0.99953379	0.00000420	0.003204031
7	1.19619730	-6.67678604	0.97191438	0.00000053	0.192314213
8	5.43696059	-2.43602275	0.36484961	0.00000007	3.545203868
9	7.81263957	-0.06034378	0.02477143	0.00000001	0.436949824
10	7.872001323	-0.00098202	0.01627374	1.00000002	0.007598169

final lambda 1	**7.872983342**
lambda 2	1

Q1 Part 3

For an approximate eigenvector x

$Ax \simeq \lambda x$

$x\lambda \simeq (Ax)$

Applying the Least Square Solution method:

$x^T x\lambda \simeq x^T (Ax)$

The term $x^T x$ is a scalar therefore divide both sides by $x^T x$.

$\boldsymbol{\lambda \simeq (x^T Ax) / (x^T x)}$ Rayleigh Quotient

Q1 Part 4

The **Inverse Shifted Power Method** converges to the Eigenvalue of matrix A which is closest to the shift value α.

Theoretical Rate of Convergence for Inverse Shifted Power Method = $\left| \dfrac{1/(\lambda_2 - \alpha)}{1/(\lambda_1 - \alpha)} \right|^k$

Where λ_1 is the eigenvalue of matrix A which is closest to the shift value α.

The **Shifted Power Method** Converges to the Dominant Eigenvalue $(\lambda - \alpha)$ of the matrix $(A - \alpha I)$.

Theoretical Rate of Convergence for Shifted Power Method = $\left| \dfrac{(\lambda_b - \alpha)}{(\lambda_a - \alpha)} \right|^k$

where $(\lambda_a - \alpha)$ is the dominant eigenvalue of the matrix $(A - \alpha I)$,
and $(\lambda_b - \alpha)$ is the next most dominant eigenvalue of matrix $(A - \alpha I)$.

Inverse Shifted Power Method Convergence Rate:

Matrix A : $\lambda_1 = 3$, $\lambda_2 = -1.5$, $\lambda_3 = 0.8$, $\lambda_4 = 0.5$, $\lambda_5 = 0.2$

Choose α = 2.5 which is closest to the dominant eigenvector of matrix A

Convergence Rate = $\left| \dfrac{1/(\lambda_2 - \alpha)}{1/(\lambda_1 - \alpha)} \right|^k = \left| \dfrac{1/(-1.5 - 2.5)}{1/(3 - 2.5)} \right|^k = \left| \dfrac{1/(-4.0)}{1/(0.5)} \right|^k = (1/8)^k$

Shifted Power Method Convergence Rate :

Matrix A : $\lambda_1 = 3$, $\lambda_2 = -1.5$, $\lambda_3 = 0.8$, $\lambda_4 = 0.5$, $\lambda_5 = 0.2$

$(\lambda_1 - \alpha) = (3 - 2.5) = 0.5$

$(\lambda_2 - \alpha) = (-1.5 - 2.5) = -4$ Dominant Eigenvalue for matrix $(A - \alpha I)$

$(\lambda_3 - \alpha) = (0.8 - 2.5) = -1.7$

$(\lambda_4 - \alpha) = (0.5 - 2.5) = -2$

$(\lambda_5 - \alpha) = (0.2 - 2.5) = -2.3$ Next most Dominant Eigenvalue for matrix $(A - \alpha I)$

Convergence Rate = $\left| \dfrac{(\lambda_b - \alpha)}{(\lambda_a - \alpha)} \right|^k = \left| \dfrac{-2.3}{-4} \right|^k = (0.575)^k$

The Shifted Power Method convergence depends upon the distance between two shifted eigenvalues.

The Inverse Shifted Power Method is much better because when the distance between the dominant eigenvalue and the shift value α is very close, the magnitude of the denominator is much bigger than that of the numerator in the convergence formula.

Q1 Part 5

The Rayleigh Quotient Iteration with updated shift value α after every iteration.

Step 1

Choose an intial guess column vector $\mathbf{x_o}$

Step 2

Choose a shift value $\boldsymbol{\alpha_1}$ close to the dominant eigenvalue $\boldsymbol{\lambda_1}$.

Step 3

Using matrix A and shift value $\boldsymbol{\alpha_1}$ **,** calculate matrix (A - $\boldsymbol{\alpha_1}$ **I) .**

Step 4

Perform one iteration of the Power Method $\mathbf{x_{new}}$ = (A - $\boldsymbol{\alpha_1}$ **I)** $\mathbf{x_o}$

Step 5

Using matrix (A - α_1 I) and column vector x_{new} calculate approx dominant eigenvalue $\boldsymbol{\lambda_1}$ using Raleigh Quotient.

Step 6

Update the shift value α_1 to a new shift value $\alpha_2 = \lambda_1$. Update $x_o = x_{new}$.

Step 7

Using matrix A and the updated shift value $\boldsymbol{\alpha_k}$ **,** calculate matrix (A - $\boldsymbol{\alpha_k}$ **I) .**

Step 8

Perform one iteration of the Power Method $\mathbf{x_{new}}$ = (A - $\boldsymbol{\alpha_k}$ **I)** $\mathbf{x_o}$

Step 9

Using matrix (A - α_k I) and column vector x_{new} calculate approx dominant eigenvalue $\boldsymbol{\lambda_1}$ using Raleigh Quotient.

Step 10

Repeat steps 6 thru 9 until convergence.

The **algorithm is stable because one condition of the shifted inverse power method is that $\alpha \neq \lambda$.**

If λ is an eigenvalue of matrix A and $\alpha \neq \lambda$, then (A - α I) is invertible if α is not an eigenvalue of matrix A and $1/(\lambda - \alpha)$ is an eigenvalue of $(A - \alpha I)^{-1}$.

MatLab Code

```
% Seth Boren
% Approximate the most DOMINANT EIGENVALUE using POWER METHOD
% Power Method using the SCALING FACTOR to find the Dominant Eigenvalue
% Input Matrix A
A = [ 1 1 1; 1 2 3; 1 3 6];
% Initial guess for eigenvector x
x = [ 1 ; 1 ; 1 ];
% set tolerance that will end iteration below
tolerance = 0.000001;
lambdabefore = 0;
%Initial "Normalized Difference" value for tolerance check
%Normalized Difference in the successive approximation of eigenvalue lambda
Normalized_Difference = 1;
% SF = scaling factor = "Infinity Norm of x" = 1
% Set initial value of scaling factor to 1
SF = 1.0;
% BEGIN LOOP
% Collect 1st 10 values of eignenvalue lamda
lambda_iteration = 1;
lambda_collect = zeros(10,1);
collect_tally = 1;
format long
while Normalized_Difference > tolerance
x_new = SF * A * x;
%Scale the eigen vector x so that "Infinity Norm of vector x"is equal to 1
I_Norm_x = norm(x_new, inf);
x_new = [ ((x_new(1,1))/I_Norm_x) ; ((x_new(2,1))/I_Norm_x);((x_new(3,1))/I_Norm_x) ];
% ***********************************************
% Scaling factor approaches eigenvalue lambda
% ***********************************************
lambda = I_Norm_x;
% "Normalized Difference" in successive approximations of lambda
 Normalized_Difference = (abs(lambda - lambdabefore))/(abs(lambdabefore));
% Collect 1st 10 iteration values for lambda
    if (collect_tally <=  10)
        lambda_collect(collect_tally,1) = lambda;
        collect_tally = collect_tally + 1;
    end
% lambdabefore is set to the previous iteration value of lamdba
lambdabefore = lambda;
% vector x is set to the previous iteration value of x_new
x = x_new;
end
final_eigenvector  = x_new;

%final_lambda = eigenvalue
final_lambda = lambda;
final_lambda
% 1st 10 iteration values of lambda
lambda_collect
final_eigenvector

%eigen_value_list = eig(A);
%eigen_value_list

%[V,D] = eig(A);
%eigen_vector_list = V;
%eigen_vector_list
%D
```

Matlab Code

```
%Seth Boren
% Approximate the most DOMINANT EIGENVALUE using POWER METHOD
% Power Method with "RAYLEIGH QUOTIENT" to find the Dominant Eigenvalue
% Input Matrix A
A = [ 1 1 1; 1 2 3; 1 3 6];
% Initial guess for eigenvector x
x = [ 1  ; 1  ; 1 ];
% set tolerance that will end iteration below
tolerance = 0.000001;
lambdabefore = 0;
%Initial "Normalized Difference" value for tolerance check
%Normalized Difference in the successive approximation of eigenvalue lambda
Normalized_Difference = 1;
% SF = scaling factor = "Infinity Norm of x" = 1
% Set initial value of scaling factor to 1
SF = 1.0;
% BEGIN LOOP
% Collect 1st 10 values of eignenvalue lamda
lambda_iteration = 1;
lambda_collect = zeros(10,1);
collect_tally = 1;
format long
while Normalized_Difference > tolerance
x_new = SF * A * x;
%Scale the eigen vector x so that "Infinity Norm of vector x"is equal to 1
I_Norm_x = norm(x_new, inf);
x_new = [ ((x_new(1,1))/I_Norm_x) ; ((x_new(2,1))/I_Norm_x);((x_new(3,1))/I_Norm_x) ];
% ***********************************************
% Calculate the eigenvalue lambda using the Rayleigh Quotient
% "More Efficient than just the Power Method with Scaling"
% ***********************************************
xt = [ (x_new(1,1)) (x_new(2,1)) (x_new(3,1)) ];
lambda = (xt*A*x_new)/(xt*x_new);
% "Normalized Difference" in successive approximations of lambda
 Normalized_Difference = (abs(lambda - lambdabefore))/(abs(lambdabefore));
% Collect 1st 10 iteration values for lambda
    if (collect_tally <=  10)
        lambda_collect(collect_tally,1) = lambda;
        collect_tally = collect_tally + 1;
    end
% lambdabefore is set to the previous iteration value of lamdba
lambdabefore = lambda;
% vector x is set to the previous iteration value of x_new
x = x_new;
end
final_eigenvector  = x_new;

%final_lambda = eigenvalue
final_lambda = lambda;
final_lambda
% 1st 10 iteration values of lambda
lambda_collect
final_eigenvector

%eigen_value_list = eig(A);
%eigen_value_list
%[V,D] = eig(A);
%eigen_vector_list = V;
%eigen_vector_list
%D
```