CPSC 302 - Assignment 6

Tristan Rice, q7w9a, 25886145

Discussed with William Qi.

1.

Omitted

2.

2.a

$$r_0 = 1, r_1 = 2, r_n = 1$$

Thus, we know the eigen values are located in the union:

$$|z - 2| \leq 1 \cup |z - 2| \leq 2$$

2.b

Yes, you can narrow this down. Since the matrix is symmetric, we know all the eigenvalues will be real.

2.c

As the iteration number goes up, the convergence rate approaches 1 i.e. it gets slower and slower as time goes on. The amount of relative difference between one iteration and the next goes to 1.

At $k = 1000$, we get an error of $8.8816 * 10^{-16}$.

```
e =

    3.9190


v =

    0.1201
   -0.2305
    0.3223
   -0.3879
    0.4221
   -0.4221
    0.3879
   -0.3223
    0.2305
   -0.1201
```

2.d

The fastest subdiagonal to reduce was the last $(n-1)$ one. The first subdiagonal was initially fast, but then the others caught up to it and was reduced last.

Eigenvalues computed

```
3.9190
3.6825
3.3097
2.8308
2.2846
1.7154
1.1692
0.6903
0.3175
0.0810
```

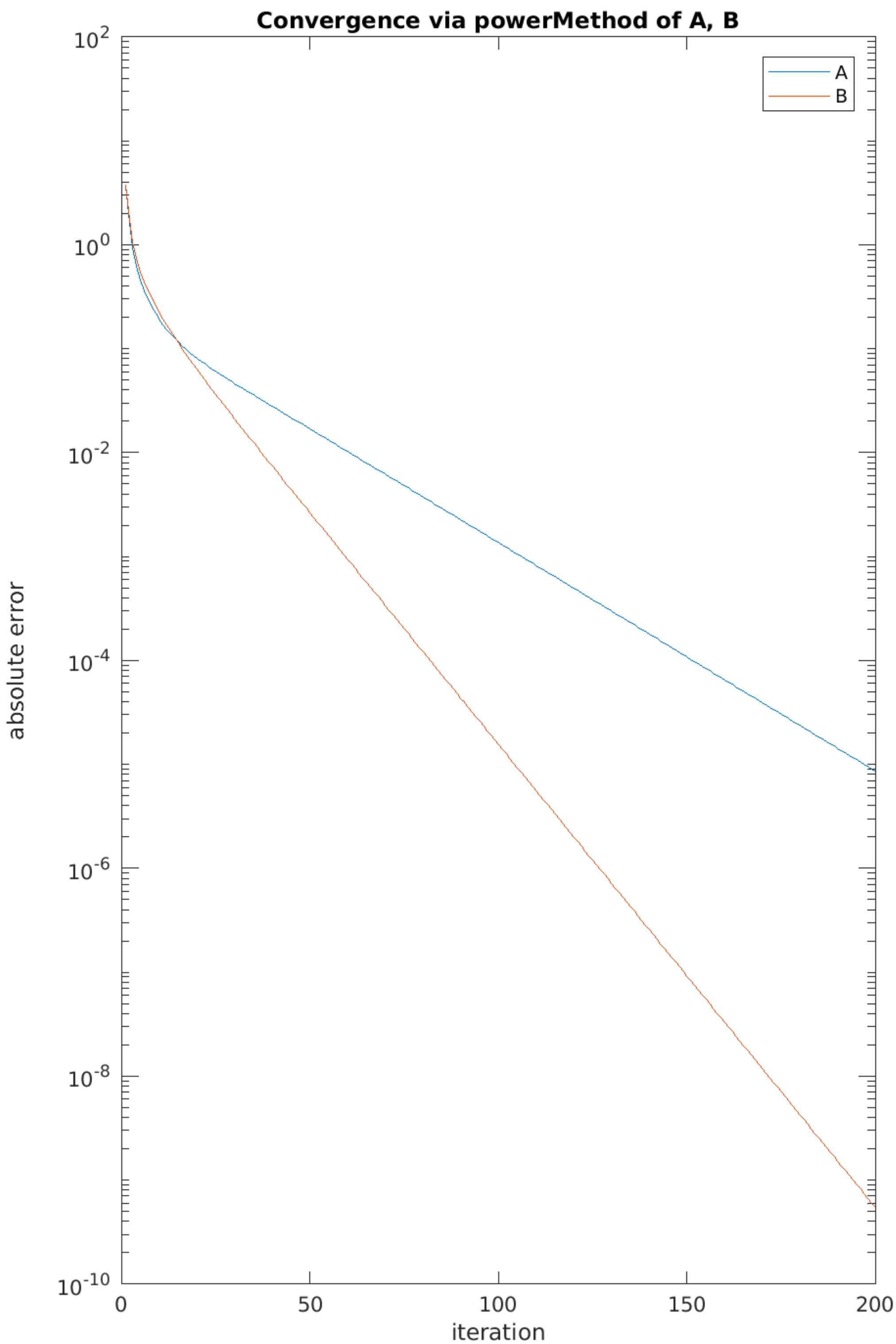
2.e

Eigenvalue 3.9190, eigenvector:

```
ans =

    0.1201
   -0.2305
    0.3223
   -0.3879
    0.4221
   -0.4221
    0.3879
   -0.3223
    0.2305
   -0.1201
```

3.

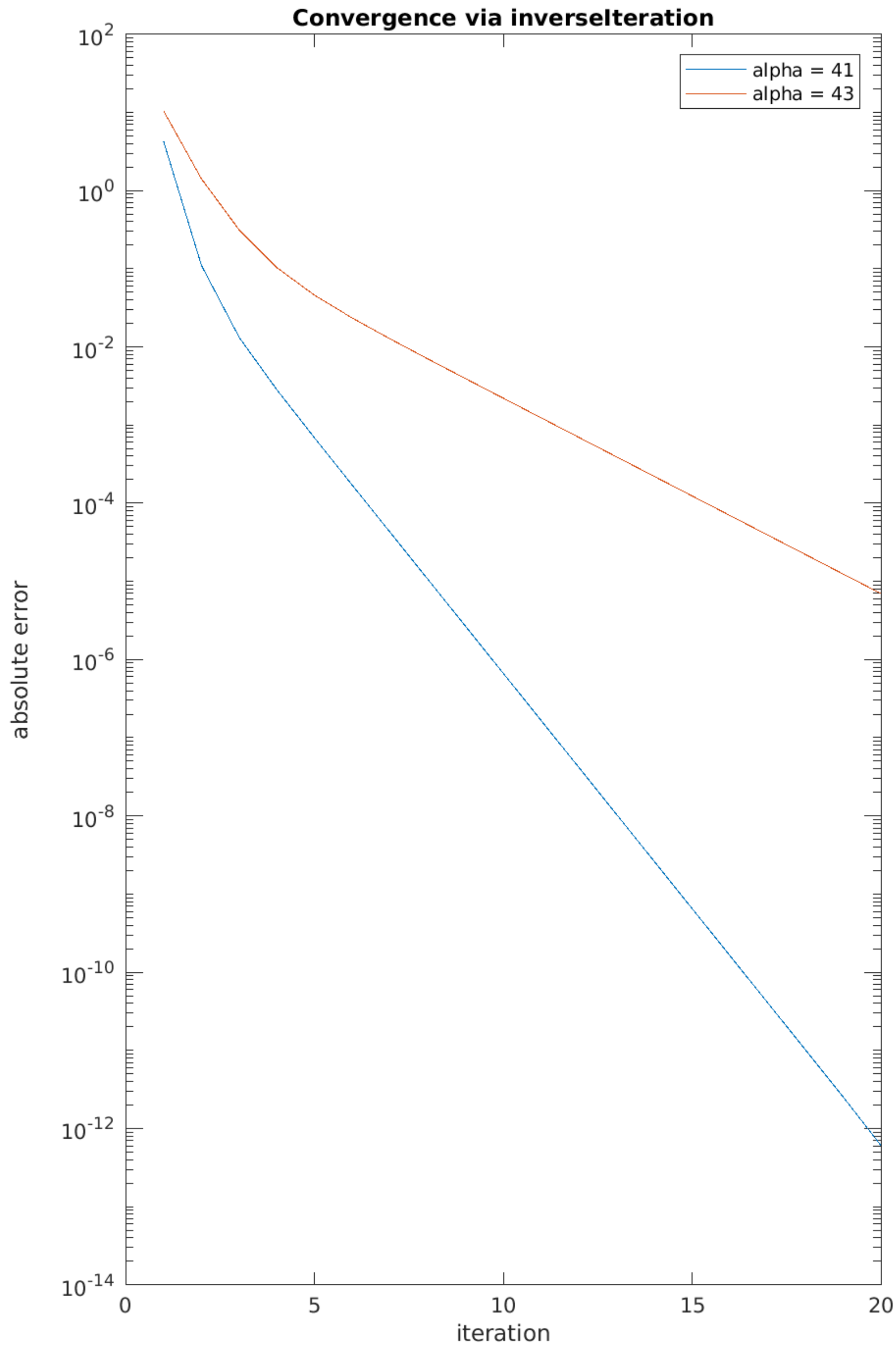
3.a


Convergence via powerMethod of A, B
A
B
absolute error
iteration
10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
0
50
100
150
200

```
Aconvergence =

   -0.0253


Bconvergence =

   -0.0513
```

If we look at the $\log(\frac{\lambda_2}{\lambda_1})$ values we see that B is twice that of A (2.0260). Those convergence rates match the graphed differences.

3.b


Convergence via inverseIteration
alpha = 41
alpha = 43
absolute error
iteration
10^2
10^0
10^-2
10^-4
10^-6
10^-8
10^-10
10^-12
10^-14
0
5
10
15
20

We see much faster iteration using inverse iteration than with the power method. We see that an eigenvalue of 41 converges much faster than an eigenvalue of 43 since it's closer to the dominant eigenvalue–40.

For Rayleigh quotient it doesn't converge to the largest eigenvalue since inverse iteration is only converging to the largest eigenvalue if the shift is closest to the largest eigenvalue. It ends up converging to a smaller eigenvalue.

4.

Omitted

```
n = 10;
A = full(gallery('tridiag', n, -1, 2, -1));
eig1 = 2 - 2 * cos((n * pi)/(n+1))
v = rand(n,1)*2;

% 2.c
[e, v] = powerMethod(A, v, eig1)

% 2.d
e = qrIteration(A)

% 2.e
% take first eigenvalue and compute the eigenvector
v = inverseIteration(A, rand(n, 1)*2, e(1))

function [e, v] = powerMethod(A, v, exact)
  for i = 1:1000
    v_old = v;
    vh = A * v;
    v = vh/norm(vh);
  end
  e = v' * A * v;
  e_old = v_old' * A * v;
  convergence = abs(exact - e)/abs(exact - e_old)
  err = exact - e
end

function e = qrIteration(A)
  err = 1;
  while err > 10^-6
    err = norm(A - diag(diag(A)), 'fro')
    [Q, R] = qr(A);
    A = R * Q
  end
  e = diag(A)
end

function v = inverseIteration(A, v, shift)
  m = size(A,1);
  I = eye(m);
  for i = 1:1000
    vh = (A - shift * I) \ v;
    v = vh/norm(vh);
  end
end
```

```
n = 40;
u = [1:n]; v = [1:n-2,n-2,n];
M = rand(n,n);
[Q,R] = qr(M);
A = Q*diag(u)*Q';
B = Q*diag(v)*Q';

q3a(n, A, B);
q3b(n, A, B);

function q3a (n, A, B)

v = rand(n, 1);
x = 1:200;
Ae = powerMethod(A, v);
Be = powerMethod(B, v);

Aeigs = eigs(A);
Aconvergence = log(Aeigs(2)/Aeigs(1))
Beigs = eigs(B);
Bconvergence = log(Beigs(2)/Beigs(1))

f = figure();
semilogy(x, Ae, x, Be);
title('Convergence via powerMethod of A, B');
legend('A', 'B');
xlabel('iteration');
ylabel('absolute error');
saveas(f, 'q3.png');

end

function [e, v] = powerMethod(A, v)
  es = eigs(A);
  exact = es(1);
  e = [];
  for i = 1:200
    v_old = v;
    vh = A * v;
    v = vh/norm(vh);
    e(i) = abs(v' * A * v - exact);
  end
end

function q3b (n, A, B)

v = rand(n, 1)*2;
x = 1:20;

a = 41;
b = 43;
ea = inverseIteration(A, v, a);
eb = inverseIteration(A, v, b);

rqe = rayleighQuotientIteration(A, v)

f = figure();
semilogy(x, ea, x, eb);
title('Convergence via inverseIteration');
legend(sprintf('alpha = %d', a), sprintf('alpha = %d', b));
xlabel('iteration');
ylabel('absolute error');
saveas(f, 'q3b.png');

end

% inverseIteration computes using the inverse iteration method.
function e = inverseIteration(A, v, shift)
  es = eigs(A);
  exact = es(1);
  e = [];
  m = size(A,1);
  I = eye(m);
```

```
  for i = 1:20
    vh = (A - shift * I) \ v;
    v = vh/norm(vh);
    e(i) = abs(v' * A * v - exact);
  end
end

% rayleighQuotientIteration computes the errors using the rayleigh quotient
% iteration method.
function e = rayleighQuotientIteration(A, v, shift)
  es = eigs(A);
  exact = es(1);
  e = [];
  m = size(A,1);
  I = eye(m);

  % normalize v
  v = v/norm(v);

  % use eigenvalue as shift
  shift = v' * A * v

  for i = 1:10
    vh = (A - shift * I) \ v;
    v = vh/norm(vh);
    shift = v' * A * v
    e(i) = abs(shift - exact);
  end
end
```