Bruce Campbell OSU MATH 5603 HW #6

Problem 5

5.3.36

```
clear all;
A = [1 1 1 ; -1 9 2 ; 0 -1 -2];

%q= [.1 .1 .1]';
q= [ 1 1 1]';
%q =[-1.1649e-01 -9.8888e-01 9.2460e-02]'; % This is an eigenvector of A
%q =[-1.1e-01 -9.8e-01 9.2e-02]'; % This is close to the first eigenvector of A
iterate(:,1)=q;
for j=1:10
    % Rayleigh quotient note ' takes complex conjugate if necessary
    rho =(q' * A *q) / ( q'*q );
    lamda_approx(j) = rho;
    % As q gets closer to v this becomes ill conditioned - expect warnings
    % cond(A-rho*eye(3))
    B=inv(A-rho*eye(3)); % Inverse shift
    q=B*q;
    [bgst,index] = max(abs(q));
    scale_factor(j+1) = q(index(1));
    q = q/scale_factor(j+1);
    iterate(:,j+1 ) =q;
end
```

```
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
2.309393e-18.
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
2.309393e-18.
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
2.309393e-18.
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
2.309393e-18.
```

```
q
```

```
q = 3×1
   1.0000e+00
   1.3774e-01
  -4.4529e-02
```

```
A*q
```

```
ans = 3×1
   1.0932e+00
   1.5058e-01
```

```
  -4.8679e-02
```

```
rho
```

```
rho =
   1.0932e+00
```

```
A*q - rho * q
```

```
ans = 3×1
            0
            0
   6.9389e-18
```

What we see from above is that we found an eigenvecor and that the final rho is it's eigenvalue

```
[V,D] =eig(A)
```

```
V = 3×3
  -1.1649e-01   9.8968e-01  -2.6661e-01
  -9.8888e-01   1.3632e-01  -1.9952e-01
   9.2460e-02  -4.4070e-02   9.4293e-01
D = 3×3
   8.6952e+00            0            0
            0   1.0932e+00            0
            0            0  -1.7884e+00
```

What we see here is we found the second eigenvalue eigenvector pair

```

format long e;
D(2,2)
```

```
ans =
     1.093208141958055e+00
```

```
lamda_approx(10)
```

```
ans =
     1.093208141958056e+00
```

```
D(1,1) -lamda_approx(10)
```

```
ans =
     7.601991344675469e+00
```

```
format shortE;
v = V(:,1)'
```

```
v = 1×3
  -1.1649e-01  -9.8888e-01   9.2460e-02
```

```
[bgst,index] = max(abs(v));
scale_factor=v(index(1));
v = v /scale_factor'
```

```
v = 1×3
   1.1780e-01   1.0000e+00  -9.3500e-02
```

```
ratios=zeros(9,1);
for j=1:9
    ratios(j) = norm( iterate(:,j+1) -v) / norm(iterate(:,j) - v);
end
table(ratios')
```

ans = 1×1 table

...

	Var1							
1	7.1070e-01	8.4226e-01	9.8733e-01	9.9981e-01	1.0000e+00	1.0000e+00	1	1

```
abs(D(2,2)/D(1,1))
```

ans =
1.2573e-01

```
errors=zeros(9,1);
for j=1:9
    errors(j) = norm( iterate(:,j) -v) ;
end
ediff = diff(errors)';
table(ediff)
```

ans = 1×1 table

	ediff							
1	-7.0403e-01	-2.7281e-01	-1.8457e-02	-2.6923e-04	-1.9599e-07	-6.6835e-14	0	0

We see that we have even faster convergence than the other methods we're tried. The large values for scale_factor and the poor warnings from the poor conditioning of $A - \rho I$ caused us to pause and think. We experimented with several starting q and notices we got the second eigenvector for the starting point of $(1, 1, 1)$. When we ran the code with the q close to the first egenvector as a starting point it converges very fast to the first eigenvector.