Dobbs4

```
format long
A=[-2 -7 5 6 5 4 4 -6;
    10 3 1 3 -4 7 6 1;
    -2 6 -4 7 10 -2 0 7;
    -10 5 -1 7 9 7 -8 -8;
    -10 5 -10 -5 4 -7 1 -1;
    -5 -6 -2 -1 8 2 9 -7;
    -6 6 -10 -5 10 0 9 -10;
    7 9 0 -1 9 -8 -7 -3];
lambda_eig = eig(A)
```

```
lambda_eig = 8×1 complex
  1.220338254800153 +19.266539064109597i
  1.220338254800153 -19.266539064109597i
-11.367130216312791 + 0.000000000000000i
 13.778857133901147 + 0.000000000000000i
  6.455431635196083 + 6.154333963016136i
  6.455431635196083 - 6.154333963016136i
 -0.881633348790417 + 2.897934266483250i
 -0.881633348790417 - 2.897934266483250i
```

```
v = [0;0;0;1;0;0;0;0];
lambda_ray = rayleigh_quotient_iteration(A,v);
```

```
lamb =
   7.088064363685821
lamb =
  11.536611500121253
lamb =
  16.145708517134260
lamb =
  14.584139431758791
lamb =
  13.843982373194905
lamb =
  13.779128191212397
lamb =
  13.778857133064740
lamb =
  13.778857133901154
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND = 9.950486e-18.
lamb =
  13.778857133901152
Warning: Matrix is singular to working precision.
lamb =
   NaN
```

this input v converges to the fourth eigenvalue / eigenvector pair

it takes 8 iterations to converge to 13 decimal places of accuracy

That's SO much better than the ~500 iterations via power iteration

```matlab
function lamb = rayleigh_quotient_iteration(A,v)
    [m,n]=size(A);
    I = eye(m,n);
    lamb = v' * A * v;
    for k=1:10
        w = (A-(lamb.*I)) \ v;
        v = w./norm(w);
        lamb = v' * A * v
    end
end
```