```
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
```