1 Problem Description

In this project I have investigate power methods for determining the eigensystem of the matrix: A = (1/2)I +B+B^T Twith B being a random 50×50 matrix with entries drawn from the normal distribution.
I have used eigs command in Matlab
to compute the 2 smallest and 2 largest eigenvalues of A and put in the results. Using two method I have calculated largest and smallest eigenvalues of A and their associated eigenvectors with initial approximation
x0 = 1, and repeat experiment for 25 times.
I have find eigenvector error put in table for both methods as well as averahe time to find lambda for both methods.

2 Results

In this project I have calculated eigen values of matrix A = 0.5I + B + B^T, with B being a random 50×50 matrix with entries drawn from the normal distribution.
Firstly, there are two largest and smallest eigen values calculate and reported (using inbuilt MATLAB function.
19.6144(largest)
18.1031(largest)
-0.0972 (smallest)
0.3190 (smallest)

Now writing the information in the table:

	Lambda time (average time for 25 iteration)	avgEigenvectorerror
Largest	0.00035041 (power method)	9.917821e-07
smallest	0.0052892(inverse power)	5.712826e-07

It looks like power method is more efficient than inverse power method. Power method converges slower, but it is faster. On the other hand, Inverse power method is slow but converge to solution in less iteration than power method. There is more error in eigen vector in power methods and it is less in inverse power method.

3 Collaboration

No collaboration on this project

4 Academic Integrity

On my personal integrity as a student and member of the UCD community, I have not given nor received any unauthorized assistance on this assignment.

5 Appendix

Attached code

```
% power methods
function [eig_val, eig_vec, time, error] = power_method(A, x0, tol, max_iters)
    tic;
    x = x0;
    for k = 1:max_iters
        y = A*x;
        eig_val = norm(y, 2);
        eig_vec = y / eig_val;
        error = norm(A*eig_vec - eig_val*eig_vec, 2);
        if error < tol
            break;
        end
        x = eig_vec;
    end
    time = toc;
end

% Inverse power method function

    function [eig_val, eig_vec, time, error] = inverse_power_method_lu(A, x0, mu, tol, max_iters)
    tic;
    [L, U] = lu(A - mu*eye(size(A)));
    x = x0;
    for k = 1:max_iters
        y = U \ (L \ x);
        eig_vec = y / norm(y, 2);
        eig_val = eig_vec' * A * eig_vec;
        error = norm(A*eig_vec - eig_val*eig_vec, 2);
        if error < tol
            break;
        end
        x = eig_vec;
    end
    eig_val = eig_val + mu;
    time = toc;
end


%Code to find error and time
n = 50;
```

```matlab
B = randn(n);
A = (1/2) * eye(n) + B + B';
[V, D] = eigs(A, 2, 'largestabs');
ground_truth_eig_val_1 = D(1,1);
ground_truth_eig_val_2 = D(2,2);
ground_truth_eig_vec_1 = V(:,1);
ground_truth_eig_vec_2 = V(:,2);
num_experiments = 25;
x0 = ones(n, 1);
tol = 1e-6;
max_iters = 1000;
power_method_eig_val_1 = zeros(num_experiments, 1);
inverse_power_method_eig_val_1 = zeros(num_experiments, 1);
power_method_time_1 = zeros(num_experiments, 1);
inverse_power_method_time_1 = zeros(num_experiments, 1);
power_method_error_1 = zeros(num_experiments, 1);
inverse_power_method_error_1 = zeros(num_experiments, 1);

for i = 1:num_experiments

   [power_method_eig_val_1(i), power_method_eig_vec_1, power_method_time_1(i),
power_method_error_1(i)] = ...
      power_method(A, x0, tol, max_iters);


   mu = ground_truth_eig_val_1 + 1;
   [inverse_power_method_eig_val_1(i), inverse_power_method_eig_vec_1,
inverse_power_method_time_1(i), inverse_power_method_error_1(i)] = ...
      inverse_power_method_lu(A, x0, mu, tol, max_iters);
end
avg_power_method_time_1 = mean(power_method_time_1);
avg_inverse_power_method_time_1 = mean(inverse_power_method_time_1);
avg_power_method_error_1 = mean(power_method_error_1);
avg_inverse_power_method_error_1 = mean(inverse_power_method_error_1);

% Print results
fprintf('Average time for power method: %f seconds\n', avg_power_method_time_1);
fprintf('Average time for inverse power method: %f seconds\n',
avg_inverse_power_method_time_1);
fprintf('Average error for power method: %e\n', avg_power_method_error_1);
fprintf('Average error for inverse power method: %e\n', avg_inverse_power_method_error_1);
```