APPM 5610 - Homework 2

Eappen Nelluvelil

February 2, 2022

1. Show that the Hilbert matrix (see below) is positive definite. **Hint:** Use

$$\int_0^1 x^{i+j-2}\,\mathrm{d}x = \frac{1}{i+j-1}.$$

To show that the $n \times n$ Hilbert matrix is positive definite, it is enough to show that it is a Gramian matrix formed from linearly independent vectors. That is, it is enough to show that $\mathbf{H} = \mathbf{v}^*\mathbf{v}$, where $\mathbf{v}$ is a vector with linearly independent columns.

Consider the matrix $\mathbf{v}^*\mathbf{v}$, where $\mathbf{v}$ is a row vector given by

$$\mathbf{v}(t) = \begin{bmatrix} t^0 & t^1 & \dots & t^{n-1} \end{bmatrix}.$$

We see that the (i,j) entry of the matrix $\mathbf{v}^*\mathbf{v}$ is given by

$$\begin{aligned} (\mathbf{v}^*\mathbf{v})_{(i,j)} &= \int_0^1 t^{i-1}t^{j-1}\,\mathrm{d}x \\ &= \int_0^1 t^{i+j-2}\,\mathrm{d}x \\ &= \frac{1}{i+j-1}, \end{aligned}$$

which is the $(i,j)^{th}$ entry of the $n \times n$ Hilbert matrix. Since $\mathbf{v}$ is a vector whose entries are the monomials up to degree $n-1$, the columns of $\mathbf{v}^*\mathbf{v}$ are linearly independent by construction, and $\mathbf{H}$ is a positive-definite matrix.

2. Implement the power method for finding the dominant eigenvalue λ_1, $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$. Use the power method to find the dominant eigenvalue and the corresponding eigenvector for the Hilbert matrix,

$$A_{i,j} = \frac{1}{i+j-1},$$

where $i, j = 1, 2, \ldots, n$. The dominant eigenvalue is well-separated in this case.

The attached code is an implementation of the power method, which is then used to find the dominant eigenvalue λ_1 and corresponding eigenvector $\mathbf{v}_1$ for the Hilbert matrix for $n = 2, \ldots, 16$.

3. Use the power method (with any modification that you may find necessary) to find the smallest eigenvalue of the Hilbert matrix with $n = 16$. How accurate is this eigenvalue? Is this consistent with the estimate $\min_{\lambda \in \sigma(\mathbf{A})} |\lambda - \mu| \leq ||\mathbf{E}||_2$, where μ is the eigenvalue is the perturbed matrix $\mathbf{A} + \mathbf{E}$? **Note:**

We modify the power method by shifting the matrix, i.e., we apply the power iteration to the modified matrix $\widetilde{\mathbf{A}} = \mathbf{A} - \lambda_1 \mathbf{I}$, where λ_1 is the largest eigenvalue (in magnitude) of $\mathbf{A}$. Since Hilbert matrices of any order are symmetric and positive definite, the eigenvalues are positive. The largest eigenvalue (in magnitude) of $\widetilde{\mathbf{A}}$ is $\lambda_n - \lambda_1$. Upon applying the power iteration to $\widetilde{\mathbf{A}}$, in theory, we would obtain the eigenvalue $\lambda_n - \lambda_1$, from which we can obtain the smallest eigenvalue of $\mathbf{A}$.

However, in practice, the obtained eigenvalue, after rearranging to solve for λ_n, is not close to the smallest eigenvalue of $\mathbf{A}$. This is because the Hilbert matrix is ill-conditioned, which can be seen in the attached code. Beyond $n = 2$, the modified power iteration returns eigenvalues that are less and less accurate when compared to the smallest eigenvalue of the $n \times n$ Hilbert matrix.

Despite the smallest computed eigenvalues not being accurate beyond $n = 2$, the computed eigenvalues is consistent with the estimate $\min_{\lambda \in \sigma(\mathbf{H})} |\lambda - \mu) \leq ||\mathbf{E}||_2$, where $\mathbf{E} = -\lambda_1 \mathbf{I}$ and $||\mathbf{E}||_2 = \lambda_1$. This is because $|\lambda_n - \lambda_1| \leq \lambda_1$ since $\lambda_n < \lambda_1$ and $\lambda_n > 0$.

4. Assume that real symmetric matrix $\mathbf{A}$ has eigenvalues $\lambda_1 = -\lambda_2$ and $|\lambda_1| = |\lambda_2| > |\lambda_3| \geq \ldots \geq |\lambda_n|$. Suggest a modification of the power method to find corresponding eigenvectors.

   Since $\mathbf{A}$ is a real symmetric matrix, it is unitarily diagonalizable, with $\mathbf{A} = \mathbf{U\Sigma U}^T$, where $\mathbf{U}$ are the eigenvectors of $\mathbf{A}$ and $\mathbf{\Sigma} = \text{diag}(\lambda_1, \ldots, \lambda_n)$.

   If we start the power iteration with a vector $\mathbf{q}^{(0)} = a_1\mathbf{u}_1 + \ldots a_n\mathbf{u}_n$, where $\mathbf{a}_i$'s are scalars and $\mathbf{u}_i' s$ are the eigenvectors of $\mathbf{A}$, we see that

$$\begin{aligned}\mathbf{Aq}^{(0)} &= a_1\lambda_1\mathbf{u}_1 + \ldots a_n\lambda_n\mathbf{u}_n \\ &= a_1\lambda_1\left(\mathbf{u}_1 - \frac{a_2}{a_1}\mathbf{u}_2 + \sum_{j=3}^{n}\left(\frac{a_j}{a_1}\right)\left(\frac{\lambda_j}{\lambda_1}\right)\mathbf{u}_j\right)\end{aligned}$$

   Applying the power iteration to $\mathbf{A}$ with $\mathbf{q}^{(0)}$, we get an eigenvector $\mathbf{x}_1 = \widetilde{a}_1\mathbf{u}_1 - \widetilde{a}_2\mathbf{u}_2$, where $\widetilde{a}_1, \widetilde{a}_2$ are scalars.

   If we then apply the power iteration to $\mathbf{A}^2$, taking the initial vector to be $\lambda_1\mathbf{x}_1$, we get an eigenvector corresponding to $\mathbf{A}^2$ given by $\mathbf{x}_2 = \widetilde{a}_1\mathbf{u}_1 + \widetilde{a}_2\mathbf{u}_2$. This can be seen by the following computation:

$$\begin{aligned}\mathbf{A}^2\lambda_1\left(\widetilde{a}_1\mathbf{u}_1 - \widetilde{a}_2\mathbf{u}_2\right) &= \widetilde{a}_1\lambda_1\mathbf{Au}_1 - \widetilde{a}_2\lambda_1\mathbf{Au}_2 \\ &= \widetilde{a}_1\lambda_1^2\mathbf{u}_1 + \widetilde{a}_2\lambda_1^2\mathbf{Au}_2 \\ &= \lambda_1^2\left(\widetilde{a}_1\mathbf{u}_1 + \widetilde{a}_2\mathbf{u}_2\right) \\ &= \lambda_1\mathbf{x}_2.\end{aligned}$$

   Thus, applying the power iteration to $\mathbf{A}$ and $\mathbf{A}^2$ yields the eigenvectors $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively. Then, to obtain $\widetilde{a}_1\mathbf{u}_1$ and $\widetilde{a}_2\mathbf{u}_2$, the eigenvectors corresponding to λ_1 and λ_2, respectively, we can perform the following computations:

$$\begin{aligned}\widetilde{a}_1\mathbf{u}_1 &= \frac{\lambda_1\left(\lambda_1\left(\widetilde{a}_1\mathbf{u}_1 - \widetilde{a}_2\mathbf{u}_2\right)\right) + \lambda_1^2\left(\widetilde{a}_1\mathbf{u}_1 + \widetilde{a}_2\mathbf{u}_2\right)}{2\lambda_1^2}, \\ \widetilde{a}_2\mathbf{u}_2 &= \frac{\lambda_1\left(\lambda_1\left(\widetilde{a}_1\mathbf{u}_1 - \widetilde{a}_2\mathbf{u}_2\right)\right) - \lambda_1^2\left(\widetilde{a}_1\mathbf{u}_1 + \widetilde{a}_2\mathbf{u}_2\right)}{-2\lambda_1^2}.\end{aligned}$$

5. A real symmetric matrix $\mathbf{A}$ has eigenvalue 1 of multiplicity 8; the rest of the eigenvalues are less than 0.1 in absolute value. Describe an algorithm, based on the power method, for finding an orthogonal basis of the 8-dimensional eigenspace corresponding to the dominant eigenvalue. Estimate the number of necessary iterations to achieve double precision accuracy.

   We can invoke the power method on $\mathbf{A}$ 8 times, using the first 8 canonical basis vectors as the initial vectors for the iteration, respectively. This will result in us obtaining the 8 linearly independent eigenvectors that correspond to the eigenvalue of $\mathbf{A}$ with multiplicity 8. In the case that we obtain linearly dependent eigenvectors, we simply run the power iteration as many times as necessary to obtain the remaining linearly independent eigenvectors corresponding to the first eigenvalue.

   Once we have obtained the 8 linearly independent eigenvectors, we can use Gram-Schmidt orthogonalization to produce an orthogonal basis for the 8-dimensional subspace spanned by the eigenvectors.

The number of necessary iterations to obtain double precision accuracy is given by the ratio of $\left|\frac{\lambda_2}{\lambda_1}\right|^n$, where n is the current iteration of the power method. Specifically, we want to find the n such that $\left|\frac{0.1}{1}\right|^n \leq 10^{-16}$, from which we get that $n \geq 16$.

```
clc;
clear;

n_max = 16;
max_iters = 100;

actual_evals = zeros(n_max-1, 1);
dom_evals    = zeros(n_max-1, 1);
dom_evecs    = zeros(n_max-1, n_max-1);

for n = 2:n_max
    H = hilb(n);

    [V, D] = eigs(H);

    z_0 = zeros(length(H), 1); z_0(1) = 1;
    [lambda, v] = power_method(H, z_0, max_iters);

    actual_evals(n-1) = D(1, 1);
    dom_evals(n-1)    = lambda;
    dom_evecs(1:n, n-1) = v;

end

disp("Actual eigenvalues and computed eigenvalues of Hilbert matrix");
fprintf("\t\tn\t\t\t\tActual eigenvalues\t Computed eigenvalues\n");
disp([(2:n_max)', actual_evals, dom_evals]);
fprintf("\n\nCorresponding computed eigenvectors\n");
disp(dom_evecs);

% Run the modified power iteration to obtain the smallest eigenvalues of
% the Hilbert matrix

actual_smallest_evals = zeros(n_max-1, 1);
smallest_evals        = zeros(n_max-1, 1);

disp("Actual smallest eigenvalues and computed smallest eigenvalues of Hilbert
 matrix");
fprintf("\t\tn\t\t\t\tActual smallest eigenvalues\t Computed smallest
 eigenvalues\n");

for n = 2:n_max
    H = hilb(n);
    [V, D] = eigs(H, n);

    z_0 = zeros(length(H), 1); z_0(1) = 1;
    [lambda_2, ~] = power_method_inv(H, z_0, max_iters);

    disp([n, lambda_2, min(diag(D))]);
end

%
```

```
% A = diag([1, -1, 0.5]);
% [l_1, v_1] = power_method(A^2, max_iters);
% [l_2, v_2] = power_method(A^2, max_iters);
%
% disp([v_1, v_2]);

% A = rand(9);
% A = triu(A) - diag(diag(A)) + diag([1, 1, 1, 1, 1, 1, 1, 1, 1e-2]);
%
% V = zeros(length(A)-1, 1);
% D = zeros(length(A), 8);
% I = eye(size(A));
%
% for i = 1:8
%     z_0 = I(:, i);
%     [lambda, v] = power_method(A, z_0, max_iters + 1000);
%     V(i) = lambda;
%     D(:, i) = v;
% end
%
%
% disp([A*D(:, 2), V(2)*D(:, 2)]);


function [lambda, v] = power_method(A, z_0, max_iters)

    for k = 1:max_iters
        z_k = A*z_0;
        q_k = z_k/norm(z_k);
        lambda = dot(q_k, A*q_k);
        z_0 = z_k;
    end

    v = q_k;
end

function [lambda, v] = power_method_inv(A, z_0, max_iters)
    [largest_e_val, ~] = power_method(A, z_0, max_iters);

    A_tilde = A - largest_e_val*eye(size(A));

    for k = 1:max_iters
        z_k = A_tilde*z_0;
        q_k = z_k/norm(z_k);
        lambda = dot(q_k, A_tilde*q_k);
        z_0 = z_k;
    end

    lambda = lambda + largest_e_val;
    v = q_k;
end
```

```
Actual eigenvalues and computed eigenvalues of Hilbert matrix
  n    Actual eigenvalues  Computed eigenvalues
```

```
   2.000000000000000   1.267591879243998   1.267591879243999
   3.000000000000000   1.408318927123654   1.408318927123654
   4.000000000000000   1.500214280059243   1.500214280059243
   5.000000000000000   1.567050691098231   1.567050691098231
   6.000000000000000   1.618899858924339   1.618899858924339
   7.000000000000000   1.660885338926931   1.660885338926931
   8.000000000000000   1.695938996921948   1.695938996921950
   9.000000000000000   1.725882660901847   1.725882660901847
  10.000000000000000   1.751919670265177   1.751919670265178
  11.000000000000000   1.774883179499381   1.774883179499381
  12.000000000000000   1.795372059561997   1.795372059561997
  13.000000000000000   1.813830118796977   1.813830118796977
  14.000000000000000   1.830594695920393   1.830594695920394
  15.000000000000000   1.845927746153488   1.845927746153487
  16.000000000000000   1.860036442743326   1.860036442743327



Corresponding computed eigenvectors
  Columns 1 through 3

   0.881674598767944   0.827044926972009   0.792608291163764
   0.471857925532024   0.459863904365544   0.451923120901600
                   0   0.323298435244499   0.322416398581825
                   0                   0   0.252161169688242
                   0                   0                   0
                   0                   0                   0
                   0                   0                   0
                   0                   0                   0
                   0                   0                   0
                   0                   0                   0
                   0                   0                   0
                   0                   0                   0
                   0                   0                   0
                   0                   0                   0
                   0                   0                   0
                   0                   0                   0

  Columns 4 through 6

   0.767854735065807   0.748719218879095   0.733225603080613
   0.445791060462709   0.440717503243512   0.436359150069654
   0.321578294480220   0.320696869822252   0.319779114044051
   0.253438943245175   0.254311386340474   0.254885556321454
   0.209822636563631   0.211530840078965   0.212844074668574
                   0   0.181442976648769   0.183143115876329
                   0                   0   0.160939670445336
                   0                   0                   0
                   0                   0                   0
                   0                   0                   0
                   0                   0                   0
                   0                   0                   0
                   0                   0                   0
                   0                   0                   0
```

```
                0                   0                   0
                0                   0                   0

Columns 7 through 9

0.720271369439766   0.709181673545698   0.699514891247007
0.432526015446475   0.429099404025192   0.425998912750593
0.318843646420194   0.317905990066158   0.316976987916454
0.255242887149454   0.255441586730359   0.255523005939058
0.213861951108334   0.214656002709988   0.215277839514775
0.184519788905940   0.185646845839823   0.186578238283591
0.162514374457332   0.163832328894961   0.164946526309270
0.145343694357308   0.146769904688278   0.147992143235347
                0   0.133026481716317   0.134310446361981
                0                   0   0.123016712833186
                0                   0                   0
                0                   0                   0
                0                   0                   0
                0                   0                   0
                0                   0                   0
                0                   0                   0

Columns 10 through 12

0.690967022219495   0.683320332980505   0.676413810285274
0.423166915294328   0.420560375555992   0.418146135572765
0.316063689551723   0.315170440634100   0.314299742629568
0.255516792083251   0.255444491435235   0.255321987188084
0.215765222817530   0.216146230854872   0.216442076211893
0.187353973498587   0.188004207938112   0.188552035725572
0.165896666582998   0.166712975590447   0.167418826249705
0.149048683067852   0.149968894636363   0.150775661872475
0.135430727654193   0.136415432471856   0.137286595261618
0.124173014540756   0.125196312973031   0.126107640595545
0.114697309959483   0.115741728398817   0.116676703819235
                0   0.107656409926997   0.108603548511526
                0                   0   0.101607996027037
                0                   0                   0
                0                   0                   0
                0                   0                   0

Columns 13 through 15

0.670125208584176   0.664359644626561   0.659042077332539
0.415898015563803   0.413794933147330   0.411819631422644
0.313452862688510   0.312630251060571   0.311831823367739
0.255161147889428   0.254970953283049   0.254758273966824
0.216669024253455   0.216839717460734   0.216964104228668
0.189015404362822   0.189408449910796   0.189742442594352
0.168032553618079   0.168568730094388   0.169039075659429
0.151487066617891   0.152117579820088   0.152678917240549
0.138061729038196   0.138754928128587   0.139377664467761
0.126923795796249   0.127658360865445   0.128322440217573
0.117518265401015   0.118279424744030   0.118970859915509
```

```
   0.109459554540885   0.110236853081041   0.110945674034342
   0.102470633381607   0.103256551896694   0.103975532710293
   0.096346867680367   0.097135934951716   0.097859772735758
                   0   0.091721739694777   0.092446507883094
                   0                   0   0.087618467755674

Actual smallest eigenvalues and computed smallest eigenvalues of Hilbert
 matrix
  n    Actual smallest eigenvalues  Computed smallest eigenvalues
   2.000000000000000   0.065741454089335   0.065741454089335

   3.000000000000000   0.002687381641412   0.002687340355773

   4.000000000000000   0.006336016073612   0.000096702304023

   5.000000000000000   0.009521353380193   0.000003287928772

   6.000000000000000   0.011164257025210   0.000000108279948

   7.000000000000000   0.011014567912347   0.000000003493899

   8.000000000000000   0.009736752934525   0.000000000111154

   9.000000000000000   0.008209232928134   0.000000000003500

  10.000000000000000   0.006950412341298   0.000000000000109

  11.000000000000000   0.006103781087902   0.000000000000003

  12.000000000000000   0.005628391022755   0.000000000000000

  13.000000000000000   0.005433514672554   0.000000000000000

  14.000000000000000   0.005433855752781  -0.000000000000000

  15.000000000000000   0.005563536430085  -0.000000000000000

  16.000000000000000   0.005775344100157  -0.000000000000000
```

Published with MATLAB® R2021b