Math 5467 – Homework 1 Solutions

Problems:

1. Register for the Piazza site https://piazza.com/umn/spring2022/math5467 and make a post to the whole class. Tell us something about yourself (e.g., is there something new you learned to do during the pandemic), and let us know if there is something specific you would like to learn in the course.

2. Write a Python function that computes the square root of a positive number using the Babylonian method. The Babylonian method to compute $\sqrt{S}$ for $S > 0$ constructs the sequence x_n by setting $x_0 = S$ and iterating

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{S}{x_n}\right).$$

   Your code can take as input a tolerance parameter $\varepsilon > 0$, and should iterate until $|x_n^2 - S| \le \varepsilon$, and then return x_n. Test your square root function to make sure it works.

   Your function should use only basic Python programming. In particular, do not use any packages, like Numpy, Scipy, etc.

   Solution is in the Python notebook from class: Google Colab Notebook.

3. Prove that the iteration in the Babylonian method above converges quadratically to the square root of S. In particular, show that the error $\varepsilon_n = \frac{x_n}{\sqrt{S}} - 1$ satisfies

$$\varepsilon_{n+1} = \frac{\varepsilon_n^2}{2(\varepsilon_n + 1)}.$$

   From this, we get that $\varepsilon_n \ge 0$ for $n \ge 1$, and so

$$\varepsilon_{n+1} \le \frac{1}{2}\min\{\varepsilon_n^2, \varepsilon_n\}.$$

   Show that this implies that $\varepsilon_n < 1$ and $\varepsilon_{n+1} \le \varepsilon_n^2$ for n sufficiently large, which is exactly quadratic convergence.

   Proof. We will start by showing that the error $\varepsilon_n = \frac{x_n}{\sqrt{x}} - 1$ satisfies

$$\varepsilon_{n+1} = \frac{\varepsilon_n^2}{2(\varepsilon_n + 1)}.$$

   Let $n \ge 0$. Then,

$$\varepsilon_{n+1} = \frac{x_{n+1}}{\sqrt{x}} - 1 = \frac{x_n + \frac{x}{x_n}}{2\sqrt{x}} - 1 = \frac{1}{2}\frac{\frac{x_n^2}{x} + 1 - \frac{2x_n}{\sqrt{x}}}{\frac{x_n}{\sqrt{x}}} = \frac{\frac{x_n^2}{x} + 1 - \frac{2x_n}{\sqrt{x}}}{2(\varepsilon_n + 1)} = \frac{\varepsilon_n^2}{2(\varepsilon_n + 1)}.$$

   Using this equation and the fact that $\varepsilon_1 = \frac{\varepsilon_0^2}{2\sqrt{x}} \ge 0$, we have that $\varepsilon_n \ge 0$ for $n \ge 1$. Since $\varepsilon_n \ge 0$ for $n \ge 1$, we have that for $n \ge 1$,

$$\varepsilon_{n+1} = \frac{\varepsilon_n^2}{2(\varepsilon_n + 1)} \le \frac{\varepsilon_n^2 + \varepsilon_n}{2(\varepsilon_n + 1)} = \frac{\varepsilon_n}{2}.$$

Likewise, it also follows that $\varepsilon_{n+1} \leq \frac{\varepsilon_n^2}{2}$. Therefore, $\varepsilon_n \leq \frac{1}{2}\min\{\varepsilon_n, \varepsilon_n^2\}$. Since, ε_{n+1} is smaller than the minimum of both values, it must be smaller than $\frac{\varepsilon_n^2}{2}$ and $\frac{\varepsilon}{2}$. Therefore, for $n \geq 1$, $\varepsilon_n \leq \frac{\varepsilon_1}{2^{n-1}}$ which is to say that ε_n converges to 0. Therefore, for sufficiently large n, $\varepsilon_n < 1$. Furthermore, since $\varepsilon_{n+1} \leq \frac{\varepsilon_n^2}{2}$,we have that $\varepsilon_{n+1} \leq \varepsilon_n^2$ for $n \geq 1$ which is to say that for sufficiently large n, $\varepsilon_n < 1$ and $\varepsilon_{n+1} \leq \varepsilon_n^2$. Thus, the series converges quadratically. □

4. Write a Python function that computes the largest magnitude eigenvalue of a square matrix with the power iteration. The power iteration is

$$x_{n+1} = \frac{Ax_n}{\|Ax_n\|}.$$

For a diagonalizable matrix, the iteration converges to the eigenvector of A with largest magnitude eigenvalue. The eigenvalue is

$$\lambda = \lim_{n\to\infty} x_n^T A x_n.$$

Compare your function to the true eigenvector and eigenvalue for small matrices where you can compute it by hand, to check that your function works. You can either run your power iteration for a fixed (and large) number of iterations, or you can compute the residual vector

$$r_n = Ax_n - (x_n^T A x_n)x_n$$

and run the iterations until $\|r_n\| \leq \varepsilon$, where $\varepsilon > 0$ is a given tolerance parameter (which can be an argument to your function).

In this exercise you may use Numpy. Try to write your code with only one loop, over the iterations in the power method.

Solution is in the python notebook from class: Google Colab Notebook.

5. Let A be a symmetric matrix. Show that any minimizer x of

$$\min\{x^T A x \,:\, \|x\| = 1\} \tag{1}$$

is an eigenvector of A with eigenvalue $\lambda = x^T A x$, *without* using that A is diagonalizable. [Hint: Any minimizer of (1) is also a minimizer of the Rayleigh quotient

$$f(x) = \frac{x^T A x}{x^T x}$$

over $x \in \mathbb{R}^n$. Compute $\nabla f(x)$, set $\nabla f(x) = 0$, and use that $\|x\| = 1$.]

Proof. First of all, we compute $\nabla f(x)$. We have the following

$$\begin{aligned}\nabla f(x) &= \frac{x^T x\,\nabla(x^T A x) - x^T A x\,\nabla(x^T x)}{(x^T x)^2} \\ &= x^T x(A + A^T)x - (x^T A x)\cdot 2x\end{aligned}$$

where in the second equality the denominator disappears because we have $\|x\| = 1$ and $\nabla(x^T Ax) = (A + A^T)x$ was given in class. Now that we computed $\nabla f(x)$ we must set $\nabla f(x) = 0$ and derive the following

$$\begin{aligned}
& x^T x(A + A^T)x - 2(x^T Ax)x = 0 \\
\Rightarrow\ & x^T x(A + A^T)x = 2(x^T Ax)x \\
\Rightarrow\ & 1(A + A^T)x = 2(x^T Ax)x && \text{(Remember } \|x\| = 1\text{)} \\
\Rightarrow\ & (A + A)x = 2(x^T Ax)x && (A = A^T \text{ since } A \text{ is symmetric}) \\
\Rightarrow\ & 2Ax = 2(x^T Ax)x \\
\Rightarrow\ & Ax = (x^T Ax)x
\end{aligned}$$

which shows us that indeed x is an eigenvector of A and furthermore the corresponding eigenvalue is $x^T Ax$ as desired. □

6. Suppose that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$, and let $a_1, \ldots, a_n$ satisfy $0 \leq a_i \leq 1$ for all i, and

$$\sum_{i=1}^{n} a_i = k,$$

where k is an integer and $1 \leq k \leq n$. Show that

$$\sum_{i=1}^{n} \lambda_i a_i \leq \sum_{i=1}^{k} \lambda_i.$$

Proof. We observe the following:

$$\begin{aligned}
\sum_{i=1}^{n} \lambda_i a_i &= \lambda_1 a_1 + \lambda_2 a_2 + \cdots + \lambda_n a_n && \text{(by definition of summation)} \\
&\leq \lambda_1 a_1 + \lambda_2 a_2 + \cdots + \lambda_k a_k + \lambda_k a_{k+1} + \cdots + \lambda_k a_n && \text{(since } \lambda_{k+i} \leq \lambda_k \text{ for all } i\text{)} \\
&= \lambda_1 a_1 + \lambda_2 a_2 + \cdots + \lambda_{k-1} a_{k-1} + \lambda_k \left(a_k + a_{k+1} + \cdots + a_n\right) \\
&= \lambda_1 a_1 + \lambda_2 a_2 + \cdots + \lambda_{k-1} a_{k-1} + \lambda_k \left(k - a_1 - a_2 - \cdots - a_{k-1}\right) && \text{(since } \sum_{i=1}^{n} a_i = k\text{)} \\
&= (\lambda_1 - \lambda_k)a_1 + (\lambda_2 - \lambda_k)a_2 + \cdots + (\lambda_{k-1} - \lambda_k)a_{k-1} + \lambda_k k \\
&\leq (\lambda_1 - \lambda_k) + (\lambda_2 - \lambda_k) + \cdots + (\lambda_{k-1} - \lambda_k) + \lambda_k k && \text{(since } 0 \leq a_i \leq 1 \text{ for all } i \\
& && \text{and } \lambda_i - \lambda_k \geq 0 \text{ for all } i \geq k\text{)} \\
&= \lambda_1 + \lambda_2 + \cdots + \lambda_{k-1} + \lambda_k(k - (k-1)) && \text{(since there are } k-1 \text{ times } \lambda_k \\
& && \text{appears before } \lambda_k k\text{)} \\
&= \lambda_1 + \lambda_2 + \cdots + \lambda_k \\
&= \sum_{i=1}^{k} \lambda_k.
\end{aligned}$$

This is the desired result that $\sum_{i=1}^{n} \lambda_i a_i \leq \sum_{i=1}^{k} \lambda_i$. □