Homework 5 Numerical Analysis Fall 2024

Instructions:

- Due 11/04 at 6:00pm on Gradescope.
- You must follow the submission policy in the syllabus

Problem 1. Let

$$\mathbf{A} = \begin{bmatrix} \frac{3}{2} & -1 & \frac{5}{2} \\ -\frac{3}{2} & 2 & \frac{1}{2} \\ -\frac{3}{2} & 2 & -\frac{3}{2} \\ \frac{3}{2} & -1 & \frac{9}{2} \end{bmatrix}$$

By hand, compute the QR factorization of $\mathbf{A}$ using the specified algorithm. Show your work at each step.

(a) regular Gram-Schmidt

(b) modified Gram-Schmidt

Problem 2. Suppose $\mathbf{U} \in \mathbb{R}^{n \times j}$ is an orthogonal matrix and $\mathbf{a} \in \mathbb{R}^n$. Recall

$$\text{proj}_{\mathbf{U}^\perp}(\mathbf{a}) = \mathbf{a} - \mathbf{U}\mathbf{U}^\mathsf{T}\mathbf{a}.$$

For each of the following, compute the number of floating point operations used:

(a) $(\mathbf{I} - \mathbf{U}\mathbf{U}^\mathsf{T})\mathbf{a}$

(b) $\mathbf{a} - (\mathbf{U}\mathbf{U}^\mathsf{T})\mathbf{a}$

(c) $\mathbf{a} - \mathbf{U}(\mathbf{U}^\mathsf{T}\mathbf{a})$

(d) Gram–Schmidt projection (Algorithm 11.2)

(e) Modified Gram–Schmidt projection (Algorithm 11.3).

Problem 3. Recall the regular Gram–Schmidt projection of $\mathbf{a}$ onto the orthogonal compliment of the columns $\mathbf{u}_1, \ldots, \mathbf{u}_j$ of $\mathbf{U}$ is

$$\text{proj}_{\mathbf{U}^\perp}(\mathbf{a}) = \mathbf{a} - \mathbf{u}_1(\mathbf{u}_1^\mathsf{T}\mathbf{a}) - \ldots - \mathbf{u}_j(\mathbf{u}_j^\mathsf{T}\mathbf{a}).$$

(a) Implement the projection from (c) in the previous problem.

(b) Compare runtime of the original `proj_perp_GS`, the modified `proj_perp_MGS` and your new implementation.

In particular, let us generate an arbitrary orthogonal matrix $\mathbf{U}$ and vector $\mathbf{a}$.

```
n = 2000
k_max = 500
U_full,_ = np.linalg.qr(np.random.randn(n,k_max))
a = np.random.randn(n)
```

```
ks = [1,50,100,150,200,250,300,350,400,450,500]

for k in ks:
    U = U_full[:,:k]

    # your timing code here for each of the 3 algorithms
```

For each of the k values above, time how long it takes to compute the orthogonal projection of $\mathbf{a}$ onto the first k columns of $\mathbf{U}$. The matrix with the first k columns of $\mathbf{U}$ is `U[:,:k]`.

Plot all the timings on the same plot, labeling each curve and the axes.

Note that because of noise, it will help to average together several runs for each value of k.

Optionally, you can repeat this and make new plots for different values of n.

(c) What do you observe? How can you explain this, given that all of the algorithms use rough the same number of floating point operations?

Problem 4. Implement a QR algorithm using your projection method. You can do this by modifying a few lines of code in Section 11.4.3.

Add a subfigure to the plot in Section 11.4.4 showing the orthogonality of the output of QR with your new implementation. How does it compare to the regular Gram–Schmidt and the Modified Gram–Schmidt? Explain why this is the case theoretically.

Problem 5. (a) Let $x_1, \dots, x_n$ be uniformly space points from -1 to 1. You can generate this in code by `x = np.linspace(-1,1,n)`.

For $n = 100$, construct the matrix

$$\mathbf{A} = \begin{bmatrix} (x_1)^0 & (x_1)^1 & \cdots & (x_1)^4 \\ (x_2)^0 & (x_2)^1 & \cdots & (x_2)^4 \\ \vdots & \vdots & & \vdots \\ (x_n)^0 & (x_n)^1 & \cdots & (x_n)^4 \end{bmatrix}$$

(b) Plot each column of $\mathbf{A}$ against $\mathbf{x} = [x_1, \dots, x_n]$ on a single plot. Label each curve.

If your matrix $\mathbf{A}$ is a stored as a python array `A`, you can plot the i-th column using `plt.plot(x,A[:,i])`.

(c) Apply a QR factorization to $\mathbf{A}$ to obtain $\mathbf{QR}$. You are allowed to use numpy's algorithm or the ones from the lecture.

On a separate plot from (b), plot each of the columns of $\mathbf{Q}$.

(d) Explain how each column of $\mathbf{Q}$ relates to $\mathbf{x}$. In particular, write down the polynomials $p_0(x), p_1(x), \ldots, p_4(x)$ such that

$$\mathbf{Q} = \begin{bmatrix} p_0(x_1) & p_1(x_1) & \cdots & p_4(x_1) \\ p_0(x_2) & p_1(x_2) & \cdots & p_4(x_2) \\ \vdots & \vdots & & \vdots \\ p_0(x_n) & p_1(x_n) & \cdots & p_4(x_n) \end{bmatrix}.$$

(e) Change n from 100 to 1000. How do the polynomials $p_0, p_2, \ldots, p_4$ seem to relate in the two cases?

What would the polynomials look like $n \to \infty$?