[MATH2605] Homework 6

Due: Feb. 23

Problem 1

Problem 2 of Homework 5 had an error. Did you notice that? I have a fix so that you have a symmetric Hessian matrix. Let's do the problem again.

Compute a quadratic function $f(x,y)$ that passes the point $(2,3)$. At the point $(2,3)$, the function $f(x,y)$ has the gradient $(1,1)$, the maximum curvature 3 along the direction $(2,1)$, and the minimum curvature 2 along the direction $(1,-2)$.

Now, let's build a Hessian from curvature (not necessarily min. or max.) requirements.
Compute a quadratic function $f(x,y)$ that passes the point $(2,3)$. At the point $(2,3)$, the function $f(x,y)$ has the gradient $(1,1)$, the curvature 3 along the direction $(2,1)$, the curvature 2 along the direction $(1,1)$, and the curvature 1 along the direction $(3,2)$. Notice that these directions are not eigenvectors.
Hint: Let $\mathbf{H} = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$, and solve for a, b, and c.

Problem 2

We will compute $\sqrt[3]{3}$. What is a simple function $f(x)$ whose solution is $\sqrt[3]{3}$? The function $f(x)$ should not have $\sqrt[3]{\ }$.
Set up the Newton iteration $x = g(x)$ for $f(x)$. Submit $f(x)$ and $x = g(x)$.
Using $x_0 = 3$ as the initial guess, perform the iteration. Submit $x_1, x_2, \ldots, x_8$.
Using $x_0 = -0.1$ as the initial guess, perform the iteration. Submit $x_1, x_2, \ldots, x_{20}$.

Problem 3

We may start learning methods to orthonormalize a matrix. In this problem, we will learn an iterative method.

Suppose that a matrix $\mathbf{A} \in \mathbb{R}^{3\times 3}$ is given, and we want to compute an orthogonal matrix that is close to $\mathbf{A}$. In fact, the matrix closest to $\mathbf{A}$ in a matrix norm can be computed from the singular value decomposition. If the SVD is $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$, the orthogonal matrix closest to $\mathbf{A}$ is $\mathbf{U}\mathbf{V}^T$. However, singular value decomposition is expensive to compute and complex to implement. So, let's use the Newton's iteration that is simple and fast.

The idea is using $\mathbf{A}$ as the initial guess for the following Newton iteration, which hopefully converge to a nearby orthogonal matrix. Consider $\mathbf{F}(\mathbf{A}) = \mathbf{A}^T\mathbf{A} - \mathbf{I}$. Its Newton iteration is

$$\mathbf{A}_{n+1} = \mathbf{A}_n - \frac{1}{2}\mathbf{A}_n^{-T}\left(\mathbf{A}_n^T\mathbf{A}_n - \mathbf{I}\right) \tag{1}$$

where $\mathbf{A}_n^{-T} = \left(\mathbf{A}_n^T\right)^{-1} = \left(\mathbf{A}_n^{-1}\right)^T$.

Using Matlab, create a random matrix $\mathbf{A}$ by using the command *A = (rand(3,3) - 0.5)*10;*. Compute the orthogonal matrix $\tilde{\mathbf{A}} = \mathbf{U}\mathbf{V}^T$ from the singular value decomposition of $\mathbf{A}$. Submit $\mathbf{A}$ and $\tilde{\mathbf{A}}$, which is the orthogonal matrix closest to $\mathbf{A}$ in a matrix norm.

If we apply the above iteration by *A = A - 1/2*inv(A')*(A'*A - eye(3));*, we will see that $\mathbf{A}$ quickly converges to an orthogonal matrix. The value of $\mathbf{A}$ will be fixed after a few iterations. Report $\mathbf{A}^T\mathbf{A} - \mathbf{I}$ per each iteration.
After $\mathbf{A}$ is converged, compare with $\tilde{\mathbf{A}}$. Are they same?
After $\mathbf{A}$ is converged, add a small random noise by using the command *A = A + (rand(3,3)-0.5)*0.1;*. Then, $\mathbf{A}$ will no longer be orthogonal, but will be quite close to an orthogonal matrix since only small noise is added, and hence, $\mathbf{A}_n\mathbf{A}_n^T \approx \mathbf{I}$ and $\mathbf{A}_n \approx \mathbf{A}_n^{-T}$. Therefore, we simplify the iteration (1) to obtain

$$\mathbf{A}_{n+1} = \mathbf{A}_n - \frac{1}{2}\mathbf{A}_n(\mathbf{A}_n^T\mathbf{A}_n - \mathbf{I}), \tag{2}$$

which may converge if $\mathbf{A}_n \approx \mathbf{A}_n^{-T}$. Notice that this iteration does not require a matrix inverse, and therefore, much cheaper and simpler.
Using the $\mathbf{A}$ (with the small noise) as the initial guess, try the iteration (2). Does the iteration converge to an orthogonal matrix?
Add much larger noise by the command *A = A + (rand(3,3)-0.5)*10;* Using (2), does the iteration converge to an orthogonal matrix?