Assignment #8

Due Monday 10 November, at the start of class

Please read Lectures 14, 15, 16, and 17 in the textbook *Numerical Linear Algebra*, SIAM Press 1997, by Trefethen and Bau.

DO THE FOLLOWING EXERCISES FROM THE TEXTBOOK:

- **Exercise 15.1** *Do parts* (a), (b), *and* (c) *only.*

DO THE FOLLOWING ADDITIONAL PROBLEMS:

P19. *The goal of this exercise is to show that the usual matrix-vector multiplication algorithm is backward-stable when we regard the* matrix *as the input; see part* **(c)**. *Always assume axioms (13.5) and (13.7) hold. Also, for simplicity, please assume all entries are real numbers.*

(a) Fix $x \in \mathbb{R}^1$. Show that if the problem is scalar multiplication, $f(a) = ax$ for $a \in \mathbb{R}^1$, then the obvious algorithm $\tilde{f}(a) = \text{fl}(a) \otimes \text{fl}(x)$ is backward-stable.

(b) Fix $x \in \mathbb{R}^3$, a column vector. Show that if $a \in \mathbb{R}^3$, a column vector, then the obvious algorithm

$$\tilde{f}(a) = \text{fl}(a_1) \otimes \text{fl}(x_1) \oplus \Big(\text{fl}(a_2) \otimes \text{fl}(x_2) \oplus \text{fl}(a_3) \otimes \text{fl}(x_3) \Big)$$

for the inner product problem $f(a) = a^* x$ is backward-stable. (*Hint. You must choose a vector norm to finish the proof. Note that multiplication comes before addition in the order of operations.*)

A proof by induction extends the way you argued part **(b)** *to show that the obvious inner product algorithm is backward-stable in any dimension; see Example 15.1. From now on you can assume it is true.*

(c) Fix $x \in \mathbb{R}^n$. Show that if $A \in \mathbb{R}^{m \times n}$ then the obvious algorithm $\tilde{f}(A)$ for the product problem $f(A) = Ax$ is backward-stable. (*Hints. Express Ax using inner products. Do not bother with scalar entries of A or x. However, you must choose a vector norm and an induced norm.*)

(d) Fix $A \in \mathbb{R}^{m \times n}$. Explain in at least 4 sentences why the obvious algorithm $\tilde{f}(x)$ for the problem $f(x) = Ax$, over $x \in \mathbb{R}^n$, is generally ***not*** backward-stable. However, this result depends on dimension. In fact, for what m, n is this $\tilde{f}(x)$ backward-stable? (*Hints. The algorithm is the same as in part* **(c)**, *but the input is x. Use what we know for inner products.*)

P20. *This question requires nothing but calculus as a prerequisite. It simply illustrates a major source of linear systems from applications.*

(a) Consider these three equations, chosen for visualizability:

$$x^2 + y^2 + z^2 = 4$$
$$y = \cos(\pi z)$$
$$x = z^2$$

Sketch each equation individually as a surface in $\mathbb{R}^3$. (Do this by hand or in MATLAB. Precision in your sketch is not important. The goal is to have a clear mental image of a nonlinear system as a set of intersecting surfaces.) Considering where all three surfaces intersect, explain informally why there are two solutions, that is, two points $(x, y, z) \in \mathbb{R}^3$ at which all three equations are satisfied. Explain why both solutions are inside the closed box $0 \le x \le 2, -1 \le y \le 1, -2 \le z \le 2$.

(b) Newton's method for a system of nonlinear equations is an iterative, approximate, and sometimes very fast, method for solving systems like the one above. Let $\mathbf{x} = (x_1, x_2, x_3) \in \mathbb{R}^3$. Suppose there are three scalar functions $f_i(\mathbf{x})$ forming a (column) vector function $\mathbf{f}(\mathbf{x}) = (f_1, f_2, f_3)$, and consider the system

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}.$$

(It is easy to put the part **(a)** system in this form.) Let

$$J_{ij} = \frac{\partial f_i}{\partial x_j}$$

be the Jacobian matrix: $J \in \mathbb{R}^{3\times 3}$. The Jacobian generally depends on location, i.e. $J = J(\mathbf{x})$. Of course, it generalizes the ordinary scalar derivative.

Newton's method itself is

$$J(\mathbf{x}_n)\,\mathbf{s} = -\mathbf{f}(\mathbf{x}_n), \tag{1}$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{s} \tag{2}$$

where $\mathbf{s} = (s_1, s_2, s_3)$ is the *step* and $\mathbf{x}_0$ is an initial iterate. Equation (1) is a system of linear equations which determines $\mathbf{s}$, which you need to *solve* for $\mathbf{s}$, and then equation (2) moves to the next iterate.

Using $\mathbf{x}_0 = (1, -1, 1)$, write out equation (1) in the $n = 0$ case, for the problem in part **(a)**, as a concrete linear system for the step $\mathbf{s} = (s_1, s_2, s_3)$.

(c) Implement Newton's method in MATLAB to solve the part **(a)** nonlinear system. Show your code. Generate at least five iterations. Use $\mathbf{x}_0 = (1, -1, 1)$ as an initial iterate to find one solution. Find the other solution using a different initial iterate. Note that `format long` is appropriate here. Check that $\mathbf{f}(\mathbf{x}_5)$ is close to zero.

(d) In calculus you probably learned Newton's method as a memorized formula:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Rewrite equations (1), (2) for $\mathbb{R}^1$ to derive this formula.