Homework 1

Due: 4:00 PM EDT on Fri Oct 8, 2021

Instructions: The homework consists of written as well as programming exercises. Your homework is your own – while you are free to discuss the problems with other students, each of you must submit your own writeup, code, etc. There is a zero-tolerance policy regarding cheating and submission of identical work.

Submitting your homework consists of two parts:

1. A **single PDF** of your written work as well as *the output of your code* should be uploaded to Gradescope (which can be accessed via Brightspace). When uploading to Gradescope, please make sure to mark which pages contain the solutions to which problems. This saves everyone a lot of time when grading.

2. Any code that you write for the assignment must also be submitted via GitHub Classroom. You will receive an invitation URL for the assignment via email. It is *your responsibility* to setup a Github account and join the GitHub Classroom.

   The code you submit *must be easy* to execute. For example, if it is Matlab code in the file `prob1.m`, the code should run when I execute the command `matlab prob1` from the command line. If the code is in C or Fortran, for example, you must include a script which compiles and executes the code. If you do not submit code for a particular problem, you will not receive credit even if you submitted the output of your code.

---

1. (**10 points**) Can Newton's method be generalized to find complex-valued roots of polynomials with real coefficients? If so, justify your answer and write a code which implements Newton's method to find the zero of

   $$f(x) = x^2 - 2x + 2$$

   that is closest to $x = i$. If not, justify your answer.

   Bonus: Can Newton's method be even further generalized to find complex-valued roots of a function of one complex variable, i.e., can Newton's method be generalized to find the roots of a function $\varphi : \mathbb{C} \to \mathbb{C}$,

   $$\varphi(z) = u(x, y) + iv(x, y)$$

   where $z = x + iy$ and u and v are real-valued functions of the real variables x, y?

2. (**10 points**) In the quasi-Newton optimization algorithm BFGS, updates to the approximate Hessian matrix on each step are in the form of a rank-two update. If $\mathbf{H}^{(k)}$ denotes the current approximate Hessian matrix, then

   $$\mathbf{H}^{(k+1)} = \mathbf{H}^{(k)} + \mathbf{U}^{(k)} + \mathbf{V}^{(k)},$$

where $\mathbf{U}^{(k)}$ and $\mathbf{V}^{(k)}$ are rank one matrices. If $\mathbf{H}^{(k)}$ is an $N \times N$ matrix, and both $\mathbf{H}^{(k)}$ and $(\mathbf{H}^{(k)})^{-1}$ are known, what is the cost of computing $(\mathbf{H}^{(k+1)})^{-1}$ using the Woodbury Matrix Identity? Instead of forming $(\mathbf{H}^{(k+1)})^{-1}$, suppose we only instead wished to apply it. What is the cost of applying $(\mathbf{H}^{(k+1)})^{-1}$ using the Woodbury Matrix Identity? Lastly, what is the *most efficient* way to compute the next iterate $\mathbf{x}^{(k+1)}$ in the BFGS algorithm? (Assuming that we can store in memory any matrices that need to be stored.)

3. For a vector $\mathbf{u} = (u_1\, u_2\, \cdots\, u_n)^T \in \mathbb{R}^n$, the following vector norms can be defined as:

$$\|\mathbf{u}\|_1 = \sum_{i=1}^{n} |u_i|, \qquad \|\mathbf{u}\|_2 = \left(\sum_{i=1}^{n} |u_i|^2\right)^{1/2}, \qquad \|\mathbf{u}\|_\infty = \max_i |u_i|.$$

   (a) (**5 points**) Prove the following two inequalities:

$$\|\mathbf{u}\|_\infty \le \|\mathbf{u}\|_2,$$
$$\|\mathbf{u}\|_2^2 \le \|\mathbf{u}\|_1\, \|\mathbf{u}\|_\infty.$$

   For each inequality, give an example of a non-zero vector $\mathbf{u}$ for which the equality is obtained. Finally, show that

$$\|\mathbf{u}\|_\infty \le \|\mathbf{u}\|_2 \le \|\mathbf{u}\|_1$$
$$\|\mathbf{u}\|_2 \le \sqrt{n}\|\mathbf{u}\|_\infty.$$

   (b) (**5 points**) Next, let $\mathbf{A} \in \mathbb{R}^{m\times n}$ be any real-valued matrix. Show that

$$\|\mathbf{A}\|_\infty \le \sqrt{n}\|\mathbf{A}\|_2,$$
$$\|\mathbf{A}\|_2 \le \sqrt{m}\|\mathbf{A}\|_\infty,$$

   where the above matrix norms are understood to be the appropriate induced norms. For each inequality, give an example matrix $\mathbf{A}$ for which equality is attained.

4. (**10 points**) Legendre polynomials are a set of polynomials $P_0, P_1, P_2, \ldots$ which are orthogonal on the interval $[-1, 1]$ with respect to the usual inner product between two functions f and g:

$$(f, g) = \int_{-1}^{1} f(x)\, g(x)\, dx. \tag{0.1}$$

   They also satisfy the recurrence relationship

$$P_{n+1}(x) = \frac{2n+1}{n+1} x P_n(x) - \frac{n}{n+1} P_{n-1}(x)$$

   which can be initialized using $P_0(x) = 1$ and $P_1(x) = x$. The roots of P_n have important applications in numerical integration, as we will see later on in the semester.

   Write a subroutine or function which calculates the roots of P_n by first using bisection to determine n disjoint intervals on $[-1, 1]$ which contain a root, and then running Newton's method with a starting guess inside each of these intervals. Each of the roots should be calculated to an absolute precision of at least 10^{-10}. The only input to your routine should be n, and the output should be an $n \times 1$ array containing the *ordered* n roots of P_n. Print out and submit the roots of P_{16}.

5. (**10 points**) Let $\mathbf{A}$ be an $n \times n$ matrix with 2 on the diagonal, -1 on the off-diagonal, and zero elsewhere. For example, with $n = 4$,

$$\mathbf{A} = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix}.$$

Show that the function $f(\mathbf{x}) = \mathbf{x}^T \mathbf{A} \mathbf{x} + 7$ is convex. Implement BFGS to compute

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}).$$

Calculate your estimate of $\mathbf{x}^*$ to an absolute precision of 10^{-10}. Initialize BFGS with $\mathbf{x}^{(0)} = (1\,1 \cdots 1)^T$ and $\mathbf{H}^{(0)} = \mathbf{I}$, where $\mathbf{I}$ is the $n \times n$ identity matrix. For $n = 10$, provide a table of your iterates $\mathbf{x}^{(k)}$, their estimated absolute precision, and the value $f(\mathbf{x}^{(k)})$.

6. (**10 points**) Consider the following Matlab code:

```
n = 10;
A = zeros(n,n);

for i=1:n
   for j=1:n
      A(i,j) = sqrt(i*i + j*j);
   end
end

x = rand(n,1);
x = x/norm(x);
b = A*x;
y = A \b;
err = norm(x-y)
```

What is the above code doing? Explain the results (i.e. why does `err` have the value that it does?).