Homework 6
APPM 4600 Numerical Analysis, Fall 2025

Due date: Friday, October 3, before midnight, via Gradescope. **Instructor**: Prof. Becker

Revision date: 10/3/2025

Theme: multivariate contraction mapping theorem, Newton's and Broyden's methods, linear algebra review.

Instructions Collaboration with your fellow students is OK and in fact recommended, although direct copying is not allowed. The internet is allowed for basic tasks (e.g., looking up definitions on wikipedia) but it is not permissible to search for proofs or to *post* requests for help on forums such as `http://math.stackexchange.com/` or to look at solution manuals. Please write down the names of the students that you worked with. Please also follow our AI policy.

An arbitrary subset of these questions will be graded.

Turn in a PDF (either scanned handwritten work, or typed, or a combination of both) to **Gradescope**, using the link to Gradescope from our Canvas page. Gradescope recommends a few apps for scanning from your phone; see the Gradescope HW submission guide.

We will primarily grade your written work, and computer source code is *not* necessary (and you can use any language you want). You may include it at the end of your homework if you wish (sometimes the graders might look at it, but not always; it will be a bit easier to give partial credit if you include your code). For nicely exporting code to a PDF, see the APPM 4600 HW submission guide FAQ.

Background Some of the problems will use a simplified version of the GNSS/GPS problem. Suppose we have a receiver at position $\boldsymbol{x}_{\text{true}} = \mathbf{0} \in \mathbb{R}^n$ (note: n is not necessarily 3 for this problem), and we have n satellites at positions $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_n$ where $\boldsymbol{e}_i$ is the canonical basis vector in $\mathbb{R}^n$, e.g., $\boldsymbol{e}_2 = [0, 1, 0, \ldots, 0]^\top$. We suppose we have exact clocks and are able to work out the exact distance from $\boldsymbol{x}_{\text{true}}$ to each satellite, which is $r_i = \|\boldsymbol{x}_{\text{true}} - \boldsymbol{e}_i\|_2 = 1$. This leads to one formulation of a nonlinear system:

$$\text{Find } \boldsymbol{x} \in \mathbb{R}^n \text{ that satisfies } (\forall i = 1, \ldots, n) \quad \|\boldsymbol{x} - \boldsymbol{e}_i\|_2 = 1. \tag{1}$$

An alternative formulation is to square each equation:

$$\text{Find } \boldsymbol{x} \in \mathbb{R}^n \text{ that satisfies } (\forall i = 1, \ldots, n) \quad \|\boldsymbol{x} - \boldsymbol{e}_i\|_2^2 = 1. \tag{2}$$

By inspection, $\boldsymbol{x} = \mathbf{0}$ is a solution for both equations — we'll call this "solution 1". By drawing this in the case $n = 2$, you might suspect there's at least one more solution, and there is. You can find it by assuming symmetry: if $\boldsymbol{x} = [a, a, \ldots, a]^\top$ then you can work out that for $a = 2/n$, this is also a solution. We'll call this "solution 2."

Problem 1: Problem 5b from §10.1 in the book, modified. Let $\boldsymbol{x} \in \mathbb{R}^3$ and $\mathbf{G} : \mathbb{R}^3 \to \mathbb{R}^3$ defined by

$$\mathbf{G}(\boldsymbol{x}) = \begin{bmatrix} \frac{13 - x_2^2 + 4x_3}{15} \\ \frac{11 + x_3 - x_1^2}{10} \\ \frac{22 + x_2^3}{25} \end{bmatrix}, \quad \text{with domain} \quad \mathcal{D} = \left\{ \boldsymbol{x} = [x_1, x_2, x_3]^\top \in \mathbb{R}^3 \mid 0 \le x_i \le 1.5 \ \forall i = 1, 2, 3 \right\}.$$

a) Show that if $\boldsymbol{x} \in \mathcal{D}$ then $\mathbf{G}(\boldsymbol{x}) \in \mathcal{D}$.

b) Compute the Jacobian of $\mathbf{G}$ (i.e., compute the gradients of each component, and then combine them).

c) Show that $\mathbf{G}$ is a contraction with respect to the ℓ^∞ norm.

d) On $\mathcal{D}$, is $\mathbf{G}$ a contraction with respect to the ℓ^2 norm? Find a bound on the Lipschitz constant with respect to the ℓ^2 norm (on $\mathcal{D}$).

e) On $\mathcal{D}$, is $\mathbf{G}$ a contraction with respect to the ℓ^1 norm? Find a bound on the Lipschitz constant with respect to the ℓ^1 norm (on $\mathcal{D}$).

f) Does $\mathbf{G}$ necessarily have a fixed point on $\mathcal{D}$? If so, is the fixed point unique?

Problem 2: Newton's and Broyden's methods

a) Write code that implements Newton's method for a generic multivariate nonlinear system $\mathbf{F}(\boldsymbol{x}) = \mathbf{0}$. It can either require the user to supply the Jacobian, or it can calculate it automatically using automatic differentiation. *Turn in your source code; you can use any language but we recommend Python*

b) Show that you can use a third-party library that implements Broyden's method. You do not need to code Broyden's method yourself! In Python, we recommend `scipy.optimize.root(..., method=broyden1',...)` (which is Broyden's "good method", similar to what our book has but with a few extra goodies). In Matlab, you can use a third party implementation from the File Exchange, but make sure to vet the implementation. *Turn in your source code that shows you using the software*

c) For both the Newton and Broyden implementations, demonstrate that they work by solving the GNSS Equation (2) in dimension $n = 3$ with a starting guess of $\boldsymbol{x}_0 = [-.1, -.1, -.1]$.

d) Again solve Equation (2) with both methods, but do this three times, once each for $n = 1000$, $n = 2000$ and $n = 4000$, to a tolerance of 10^{-9} (i.e., a condition like $\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2 \leq 10^{-9}$)[1]. Make two plots, one plot showing the number of iterations required as a function of n (both methods on the same plot), and the other plot showing the time per iteration as a function of n (again, both methods on the same plot).

e) Discuss the results you found above. For this particular problem, which method do you recommend? *Bonus: Do you think it depends on your implementation? Profile your Newton method code to see what the slowest step is. If it's not the linear system solve, then your code is not as efficient as it could be!*

Problem 3: Fixed point iteration

a) Write code to solve Eq. (1) and Eq. (2) via the fixed point iteration. These are in the form $\mathbf{F}(\boldsymbol{x}) = \mathbf{0}$, so convert them to a fixed point equation $\boldsymbol{x} = \underbrace{\boldsymbol{x} + \eta\mathbf{F}(\boldsymbol{x})}_{\mathbf{G}(\boldsymbol{x})}$ for a scalar $\eta \neq 0$.

Turn in the code.

b) Let $\boldsymbol{x}_0 = [a, a, \ldots, a]^\top \in \mathbb{R}^n$ be the starting vector for the fixed point iteration, and we'll consider a range of 100 different a values equally spaced from -1 to 1.5. For each starting value a, run the fixed point iteration on both Eq. (1) and Eq. (2) (that is, squared and not-squared formulations), using $\eta = \frac{1}{2}$. Make a plot that indicates for which a values the fixed point iteration converges to the first fixed point, and for which a values it converges to the second fixed point (for both squared and not-squared formulations); and note that it may converge to neither (or diverge). *Details: run your fixed point iteration for at most 100 iterations, and declare that it has converged to a given fixed point $\boldsymbol{p}$ if its output $\boldsymbol{x}$ is within 10^{-6} of $\boldsymbol{p}$ in the Euclidean norm. We suggest you do this problem for $n = 5$ but any $n \geq 3$ is fine.*

c) Repeat the process for $\eta = -\frac{1}{2}$. Turn in the plot.

d) Comment on what you've observed.

Problem 4: Let $\mathbf{A} = 2\mathbf{I} + 2\boldsymbol{u}\boldsymbol{v}^\top$ be the $n \times n$ matrix where $\boldsymbol{u} \in \mathbb{R}^n$ is the all ones vector, and $\boldsymbol{v} = [1, -1, 1, -1, 1, -1, \ldots]^\top \in \mathbb{R}^n$ *Hint: in Python, use* `np.tile` *to create* $\boldsymbol{v}$. Let $\boldsymbol{b} \in \mathbb{R}^n$ be the vector with n equally spaced points starting at 0 and ending at 1, e.g., for $n = 5$ then $\boldsymbol{b} = [0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1]^\top$. Find a solution $\boldsymbol{x}$ to the equation $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ for the case $n = 10^6$, and return

[1] With `scipy.optimize.root`, do not pass in `tol` nor `xtol` since they will set a relative tolerance compared to the true solution, and since the true solution is $\mathbf{0}$, this condition will never be triggered. Instead, set the absolute tolerance via the `xatol` flag.

the mean of the solution (i.e, $\hat{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$) and the sample standard deviation (i.e., using Bessel's correction, so $\hat{\sigma} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \hat{x})^2}$). *You are allowed to use a computer for this question; also, you may want to check your answer via another method for a smaller n.*