MATH 2160, Chapter 1 Summary & Exercises

Richard M. Slevinsky

A Conversation with Slevinsky

Problems	Solutions
What is a norm?	A norm $\|\cdot\|$ is a function that is non-negative (0 only if the vector is zero), satisfies the triangle inequality, and is scalable (scaling the vector scales the norm by the absolute value of the scalar). Intuitively, it measures the "size" of a vector.
The ∞-norm of a function was defined by the supremum. What is that?	The supremum is the *least upper bound* of a set. For example, the maximum of $[-1, 1]$ is 1, but what is the maximum of $[-1, 1)$? Indeed, this set has no maximum since we can always take an element that is half the previous distance from 1 but is not 1. On the contrary, the supremum of both sets is 1.
The supremum also showed up in defining induced matrix norms.	Yep, but we were able to convert it to a maximum using the scalability of norms. This is because the set (search space) $\{x \in \mathbb{C}^n : x \neq 0\}$ is unbounded but $\{x \in \mathbb{C}^n : \|x\|_p = 1\}$ is a closed and bounded set, and thus the supremum is *attained* in the second set.
What is an inner product?	It is a function denoted $\langle \cdot, \cdot \rangle$ that has conjugate symmetry, linearity in the second argument, and positive definiteness. In $\mathbb{R}^2$ and $\mathbb{R}^3$, the inner product is the familiar dot product, which is used to measure "how much" one vector points in the direction of another. This notion is true more generally in $\mathbb{R}^n$ or $\mathbb{C}^n$ or other 2-normed vector spaces. In fact, when $\langle x, y \rangle = 0$ for some vectors x and y, then we say that they are *orthogonal*.
Why did we discuss inner products? Aren't norms enough?	Inner products are extremely important because of their relation to the 2-norm, $\|x\|_2 = \sqrt{\langle x, x \rangle}$, and the Cauchy–Schwarz inequality. This will facilitate much of our analysis, and allows us to construct really cool algorithms in ℓ^2 and L^2 spaces that simply don't exist in the more general ℓ^p and L^p spaces, $p \neq 2$.

How does a computer represent numbers?

There are many different ways to represent numbers on a computer. For numerics, floating-point numbers are the most important, so much so, that IEEE created the 754 standard for floating-point arithmetic. Floating-point numbers are just a scaled finite geometric series in base b:

$$\pm b^c \sum_{k=0}^{n} a_k b^{-k}.$$

The sign, the exponent c, and every coefficient a_k, are the data to be represented on a computer. The most convenient base to represent numbers on modern computers is binary, $b = 2$. This is because 0 and 1 are represented using precisely one bit of information.

What kinds of error should I be aware of when implementing a numerical algorithm?

There are many sources of error: measurement error, rounding error, and human error. There is also overflow and underflow to consider; this happens when a sequence of floating-point operations leads to real numbers that are larger than may be representable in so many bits. Norms are useful in measuring most sources of error on a computer.

What is conditioning?

Conditioning describes the sensitivity of a problem to perturbations in the data. A well-conditioned problem is one in which small perturbations in the input lead to small perturbations in the output. An ill-conditioned problem may map small perturbations in the input to huge disagreements in the output.

Hmm, sounds like solving an ill-conditioned problem is a bad idea. What about stability?

Stability describes the error propagation in algorithms. A stable algorithm provides the *exact* solution to a nearby problem; an unstable algorithm does not. This implies that the error of a stable algorithm scales with the precision at hand: if we double the number of digits (or bits), then we expect to double the accuracy of the output.

What is big-$\mathcal{O}$ notation?

Big-$\mathcal{O}$ notation is the mathematical equivalent of estimating the size of the moon by covering it with one's thumb (and using the underlying trigonometry). It's all about estimation, hiding constants irrelevant to the problem at hand.

Ah, I get it now.

Great, because in the next chapter, we will be using this notation quite a lot to discuss the computational costs of solving linear systems $Ax = b$ in terms of the problem's dimensions $A \in \mathbb{C}^{m \times n}$.

Neat!

And one last thing, review theorems from calculus. They're summarized in the course notes, but only briefly and without proof.

Exercises

1. Distance may be defined in terms of norms: $\mathrm{dist}_p(x,y) = \|x-y\|_p$. In $\mathbb{R}^2$: dist_2 represents euclidean distance, i.e. a bird's distance between x and y; dist_1 is the sum of the distances along coordinate axes in absolute value; and, dist_∞ is the maximum absolute value of distances along coordinate axes. Working with different norms is not only an abstract concept but also useful: we don't drive to university in the direction of smallest euclidean distance. Let $x = (1,2,3,-4,5)^\top$ and $y = (1,-1,1,0,1)^\top$. Calculate:

$$\mathrm{dist}_1(x,y), \qquad \mathrm{dist}_2(x,y), \quad \text{and} \quad \mathrm{dist}_\infty(x,y).$$

2. Matrix-vector multiplication allows us to determine the vector $b := Ax$, where $A \in \mathbb{F}^{m\times n}$, $x \in \mathbb{F}^n$, and $b \in \mathbb{F}^m$. In a language that uses *column-major ordering*, i.e. the entries of A are stored sequentially by columns $A_{1,1}, A_{2,1}, \ldots, A_{n,1}, A_{1,2}, A_{2,2}, \ldots$, an efficient algorithm for matrix-vector multiplication is:

```
function matvec(A::Matrix, x::Vector)
    m, n = size(A)
    n == length(x) || throw(DimensionMismatch(
    "second dimension of A, $n, does not match length of x, $(length(x))"))
    b = zeros(m)
    for j = 1:n
        xj = x[j]
        for i = 1:m
            b[i] = b[i] + A[i,j]*xj
        end
    end
    b
end
```

   How many additions and multiplications are required to carry out matrix-vector multiplication?

3. The function $\mathrm{vec}(\cdot)$ maps a matrix $A \in \mathbb{F}^{m\times n}$ to the vector $a \in \mathbb{F}^{mn}$ such that $A_{i,j} = a_{i+m(j-1)}$. Show that the function $\|\cdot\|_F = \|\mathrm{vec}(\cdot)\|_2$ is a matrix norm. (Indeed, it is known as the Frobenius norm.) Prove the inequality $\|AB\|_F \le \|A\|_F \, \|B\|_F$.

4. The function $\Lambda(z) := \dfrac{\Gamma(z+\frac{1}{2})}{\Gamma(z+1)}$ is a special function. Use Stirling's formula for the gamma function:

$$\Gamma(z+1) \sim \sqrt{2\pi z}\left(\frac{z}{e}\right)^z, \quad \text{as} \quad z \to \infty,$$

   to derive the leading order asymptotic behaviour of $\Lambda(z)$ as $z \to \infty$. First, conclude that:

$$\Lambda(z) \sim \left(\frac{z-\frac{1}{2}}{z}\right)^z \frac{e^{\frac{1}{2}}}{\sqrt{z}}, \quad \text{as} \quad z \to \infty,$$

   and then use the linear approximation $\log(1+x) \sim x$ as $x \to 0$ to simplify further.

   Asymptotic expansions are not unique. For example, it is also true that:

$$\Lambda(z) \sim \frac{1}{\sqrt{z+\frac{1}{4}}} \quad \text{as} \quad z \to \infty.$$

   Use JULIA to compare and contrast the pointwise relative error of this result to the result you obtain above for $z = 1, 3, 10, 30, 100, 300, 1000$. Implement $\Lambda(z)$ by the formula `exp(lgamma(z+0.5)-lgamma(z+1))`.