Incorporating MATLAB® into University Mathematics Programs

by

Alexander Stanoyevitch
Division of Mathematical Sciences
University of Guam
alex@math.hawaii.edu

Abstract: MATLAB®, an acronym for "MATrix LABoratory", is the most extensively used mathematical software in the general sciences. Other software packages, such as Maple and Mathematica are also popular, but more so in mathematics departments than in other fields. The latter two software packages work symbolically, whereas as MATLAB, in its default mode, works in floating point arithmetic, which is much faster for performing computations with sufficient accuracy for most applications. For instances where symbolic functionality is required, MATLAB offers access to the Maple kernel (which is built in), and thus can offer the best of both worlds. With its superb graphics and user-friendly interface, MATLAB is an ideal software to enhance university mathematics courses. Indeed, because of its wide use in industry and the research sciences, a university education in mathematics without MATLAB could pose a serious disadvantage to a student's potential for landing suitable employment.

We will discuss the implementation of MATLAB into a variety of courses in the university curriculum, spanning from individual use by a single faculty member to full-scale adaptation by a mathematics department in its programs. After our general implementation summary, we will present a detailed example of incorporating MATLAB in the solution of a family of problems concerning air travel logistics that could be presented in courses ranging from the precalculus-level all the way up through upper-level mathematics courses (depending on the degrees of detail and analysis that are presented). Our focus is primarily geared to undergraduate curricula.

SECTION 1: HOW I BEGAN TO WORK MATLAB® INTO MY COURSES

The story of how I started using MATLAB in my courses is somewhat atypical, and can perhaps serve as motivation for those having hesitations on making such a transition. My doctoral research was in pure mathematical analysis and at my first post at the University of Hawaii, I continued with my pure mathematical research and was teaching math courses governed by departmental syllabi that were quite traditional. There were no specific technology requirements, but instructors could use it at there own discretion, as long as there was time to cover the required topics. For many courses, the syllabi were too ambitious to allow much digression into technology. I was learning that showing the students interesting applications of the theory was a most effective motivator for them to appreciate (and thus more thoroughly study) the mathematical theories and concepts being presented. I thus began planning my lessons to include an increased emphasis on applications, while still covering all of the theory. I began using the Maple® and Mathematica® software for some research inquiries as well as in teaching demos.

I subsequently found an interesting challenge at the University of Guam when they were looking for a new faculty member who would be primarily responsible for teaching their numerical and applied mathematics courses. Ordinarily such positions are filled by candidates who completed their doctoral work in such a field, so I was eager to take on this exciting opportunity. The only problem was that the software they were using at the time was MATLAB, with which I had no experience. My initial impression was a bit negative on using this software over one of its symbolic counterparts (like Maple or Mathematica). This impression was partially fostered from the attitudes that were widely prevailing among many pure mathematical circles that symbolic software (like Maple or Mathematica) were the ONLY real tools that should be used for mathematical work. Well, my new career thus got off to a rough and busy start, having to quickly become adept with this new computing platform. As I began to learn and use MATLAB, however, I became increasingly impressed with its speed, power, ease of use, and state-of-the-art graphical capabilities. I also came to understand why MATLAB is the software of choice in nearly all of the scientific fields. Rather than using **symbolic arithmetic** (in its default setting), which works in expandable precision to keep answers exact, MATLAB works in a **floating point arithmetic** that limits the size and number of digits in its calculations. These limits conform to meet the rigors of the IEEE[1] double precision standards, which are sufficient for most practical purposes. After all, if we are computing, say a minimum eigenvector of a certain (symmetric) matrix, is it really necessary to know the number, say, 150 digits? Pure mathematicians often scoff at anything less than exactness, but in practical situations, such objections are often purely academic. In any case, MATLAB does have symbolic functionality available for instances were one (feels) it is necessary. The basic symbolic functionality is included in the Student Version of MATLAB, and it is available with the so-called Symbolic Toolbox add-on to the professional version. The Symbolic Toolbox gives MATLAB users access to the Maple kernel. Although it is tempting to work symbolically, the trade-off is that calculations are much more expensive (time-consuming). Indeed, in my first MATLAB-based textbook [Sta-05a] on the subject of numerical ordinary and partial differential equations, although symbolic functionality is discussed, its use was extremely minimal, and the text could be made essentially independent of it.

The many benefits and versatility of MATLAB began to motivate me to use it in more than just the originally intended classes in numerical and applied mathematics. I began incorporating it, to various degrees, in most all of my math classes. This semester, for example, I am making extensive use of MATLAB for the first time in a course in abstract algebra. Students these days are inkling to use their computers in any way possible to help them learn. I am finding that the incorporation of MATLAB brings my courses to higher levels in that it enables students to perform more independent research that would otherwise not have been feasible at the undergraduate level.

SECTION 2: HOW CAN YOU IMPLEMENT MATLAB INTO YOUR COURSES?

Many math departments are gravitating towards including a MATLAB component in their curriculum. Once this is done, courses can be organized in an efficient manner with some core MATLAB-based courses giving students preparation to jump into more advanced courses with a solid MATLAB foundation behind them. If you are in such a department, this section may be skipped. The next best thing would be a department with other faculty already using MATLAB in their courses. This would be a middle-ground that could eventually lead up to the first sort of curriculum structure. If you are in such a department, your colleagues can be valuable resources not only to consult with on their experiences, but to begin to organize sequences of math courses that build up and nicely complement MATLAB implementation; such courses could, for example, be made part of an “Applied Mathematics Track” that would constitute one of several tracks in which students could earn a bachelors degree in mathematics. Graduate programs are often more specialized, making it easier to build around a particular computing platform; this is why our focus is geared towards undergraduate curricula.

Now we are left with the remaining situations where no one else in your department is currently using MATLAB, and your department may or may not even have a license for it. Through discussions with other mathematics department chairs and administrators, I am finding that most will support such innovative goals in teaching relating to bringing a high-power computing platform (like MATLAB) into the curriculum. What would be required, initially, would be a classroom license for a sufficient number of computers in a computer lab. These licenses are quite affordable (go to `www.mathworks.com`, the website for MathWorks, the company that sells MATLAB, for more detailed information) and include permission for all instructors to install the software on their personal computers to use (only) for class preparations. Additionally, a few additional licenses (a minimum of, say, four or five) would need to be purchased for the student computer lab. These licenses are more expensive, but can be set up on a central server that would allow any machine to run it as long at the maximum number of licenses is not exceeded

[1] IEEE is the acronym for the Institute for Electrical and Electronics Engineers, Inc., a nonprofit professional association of more than 350,000 individual members in 150 countries. Their standards were carefully developed to help avoid some problems and incompatibilities with earlier floating point systems.

at any given time. Finally, you should instruct your student book store to order some copies of the Student Version of MATLAB, as many students will prefer to use it on their own computers (these licenses are kept affordable, last time I checked they ran under US$100).[2] You might first begin using MATLAB in an easy-to-use course such as linear algebra (it is great with matrices), or numerical analysis.

In the isolated use of any technology in any (mathematics) course, one must also address the trade-off with time needed to introduce the technology being taken from the limited time of the semester (or quarter). I have faced this issue myself in an assortment of courses where I had wanted to use MATLAB, but some (or all) of the students had no MATLAB (and some no programming) experience. Personally, I feel that when I teach my students MATLAB, it is worthwhile to teach them some at least some of the associated rudimentary programming as well. I can usually give such students an adequate preparation in two weeks of class instruction. The exact material (and homework exercises) I cover, will vary with the course, but I always cover at least the following core material from [Sta-05b][3] : Chapter 1 (MATLAB Basics), Chapter 3 (Introduction to M-files[4]), Chapter 4 (Programming in MATLAB), Sections 5.1-2 (Floating Point Arithmetic) Sections 7.1, 3, and 4 (Matrices and Linear Systems), and Appendix B (Symbolic Computations).[5] After this, by carefully including computational problems in the assignments, the students will continue to become more adept with MATLAB while they use it to help learn the material of the course proper. Many of MATLAB's built-in M-files are open source, so students can look at these to continue to learn efficient programming ideas, and modify existing programs to better meet their needs.

I should add that since MATLAB has such a wide variety of built-in functions, it is also possible to use it without having the students do any programming (some scientists use it in this way as well). Some (MATLAB experienced) instructors will write MATLAB programs themselves specifically for the course and provide them to the students as learning tools, and many will share them with other interested instructors and/or students. Web searches on free MATLAB material will reveal the vast amounts of programs and documents that are available. A good example of this is Rice University Professor John Polking's M-files `pplane` and `dfield`. Over the past decade, he has continually developed these user-friendly, high-level, graphically enhanced M-files for analyzing phase-planes and direction fields in ordinary differential equations. He makes them freely available for educational purposes, along with instructions for their use (on his web site). He has also co-written a supplementary book [Pol-03] for standard courses in ordinary differential equations that elaborates on their use, as well as many of MATLAB's built-in functions for solving ordinary differential equations. MathWorks maintains a website [Mat-05] with descriptions of an ever-expanding list of over 800 textbooks that use the MATLAB software to various degrees in an assortment of subjects.

Currently I am using MATLAB, to various extents, in most all of the courses that I am teaching. The usage of MATLAB in my courses ranges from minimal (e.g., an occasional in-class demo for a lower-level class in finite mathematics, on, say, the shipping logistics material in Section 3, or on cryptology), to higher-level involvement requiring students to write programs to solve difficult problems in the class. MATLAB proficiency also makes it easier to get students involved in capstone or interdisciplinary research/dissertation projects. Presently, for example, I have a student in my discrete structures class writing a paper on ant colony methods in combinatorial optimization (these can serve as effective heuristics in difficult problems such as the traveling salesman and vehicle routing problems). I have also made extensive use of MATLAB in an abstract algebra course. Surprisingly, even in such a symbolic and abstract subject, MATLAB's floating point system (restricted to integer arithmetic) is impressively adequate for all but the most technical applications (e.g., in professional-level security cryptography). The abstract concepts can be made more concrete with the resources of MATLAB. Long computations, that would have otherwise been unfeasible, can be carried out to glean greater insights into the subject. For example, the paper [Mak-02] shows many interesting ways to experiment with a wide variety of matrix groups and rings using MATLAB.

[2] MathWorks will even send instructors interested in using MATLAB a complimentary copy of the Student Version; see their website for details.

[3] [Sta-05b] contains the general core introduction to MATLAB culled from my larger numerical analysis textbook [Sta-05a].

[4] M-files are simply programs written for MATLAB; the programming language of MATLAB is similar to the C-family.

[5] Much of this material is of a tutorial nature, and so can be assigned as homework.

Perhaps one of the most important reasons to learn a software like MATLAB is that so many (in fact, most) interesting mathematical problems do not have an exact analytical solution, or it is not feasible to find one. In such cases, simulations can be used to obtain approximate answers to any degree of accuracy. I have used simulations extensively in an upper-level probability course, and have also recently taught a MATLAB-based special topics course in simulation using the (non-MATLAB) textbook [Ros-02]. Even when there is an exact answer that the student needs to find (in a homework problem, say), a student (or teacher) may get stuck or go in the wrong direction with a tricky problem, but a simple simulation can point them in the direction of the correct answer. Simulations provide an excellent example of the superiority of floating point arithmetic for certain computational tasks. Results improve with running large numbers of trials, and floating point arithmetic gives us this speed whereas symbolic computations would proceed at much slower pace, keeping very exact track of all of the incidental quantities. Sometimes a program can model such a complicated natural phenomena that it is helpful to have a graphically enhanced program to check that things are working as planned. MATLAB's superb graphics capabilities allow for this. It is important to realize, however, that graphical output also eats up a lot of resources, so that a good scheme might be to first write a graphically enhanced program (for a difficult model), use it to check and demonstrate the accuracy of the model, and then disable the graphics so as to be able to do longer and more repeated runs of the program. Such a project was carried out in [Sta-04], where a graphically enhanced MATLAB program was developed to analyze the most effective way to control a traffic intersection.

SECTION 3: A PARTICULAR EXAMPLE OF A FAMILY OF PROBLEMS

In this section we present details of an example of some problems relating to the logistics of air shipping. Many of the concepts we will introduce have applications in the tourism and travel industry. Most of the ideas will be motivated by a series of practical questions. The basic prerequisites are minimal, so the material can be presented in wide variety of courses. Suppose that a Pacific Rim air shipping company has connecting flights between five cities as shown in Figure 1. The ideas will work for networks of any size, but such a small one will allow us to verify the concepts geometrically.

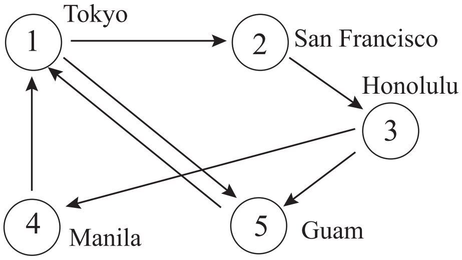

FIGURE 1: A small shipping network.

The information contained in such a network can be represented using a 5×5 **incidence matrix** A, where each of the entries of A is either 0 or 1 as determined by the following rule:

$$a_{ij} = \begin{cases} 1, & \text{if there is a direct flight from city \#}i\text{ to city \#}j \\ 0, & \text{if there is no direct flight from city \#}i\text{ to city \#}j \end{cases}.$$

Thus we obtain the incidence matrix A:

$$A = \quad \text{Rows = Origins} \quad \begin{array}{c|ccccc|} \multicolumn{6}{c}{\text{Columns = destinations}} \\ & \#1 & \#2 & \#3 & \#4 & \#5 \\ & \Downarrow & \Downarrow & \Downarrow & \Downarrow & \Downarrow \\ \#1\Rightarrow & 0 & 1 & 0 & 0 & 1 \\ \#2\Rightarrow & 0 & 0 & 1 & 0 & 0 \\ \#3\Rightarrow & 0 & 0 & 0 & 1 & 1 \\ \#4\Rightarrow & 1 & 0 & 0 & 0 & 0 \\ \#5\Rightarrow & 1 & 0 & 0 & 0 & 0 \end{array}$$

Such a spreadsheet can easily be queried as to the existence or nonexistence of a direct flight between any distinct pair of cites. Of course, the graph could tell us the same information, perhaps even more easily. But for a large network, as for example the network of all of FedEx's airport hubs, such a diagram would be a dizzying mess, and the incidence matrix would be the best way to store and access the information. If this was the only novelty of incidence matrices, they would not be worth discussing in this section. It turns out that the square of the incidence matrix tells us something quite interesting about the network (and thus furnishes a great answer to the common question students ask when first learning how to multiply matrices: "Why would anyone invent such a complicated way to do it?")

QUESTION 1: What do the entries of the matrix A^2 tell us about the shipping network?

Answer: The (i,j) entry of A^2 equals the number of ways that one can fly from (row) city #*i* to (column) city #*j* using **exactly two** flights. More generally, the entries of A^M (M a positive integer) count the number of ways to go from row city to column city using **exactly *M*** flights.

The proof of these facts is not hard (using the definition of matrix multiplication and induction), and would make a suitable exercise in a linear algebra course, but for a lower-level class, the result can just be stated and demonstrated, as we do next.

For the network of Figure 1, and its incidence matrix *A*, we compute: $B = A^2$, and $C = A^3$, and interpret the entries b_{31}, b_{12}, and c_{12}.

Using MATLAB, we enter the incidence matrix *A*, and compute *B*, *C*, and *D*.

```
>> A=[0 1 0 0 1; 0 0 1 0 0;0 0 0 1 1;1 0 0 0 0;1 0 0 0 0];
>> B=A^2,  C=A^3
```

```
→B=        1  0  1  0  0          →C=        0  1  0  1  2
           0  0  0  1  1                     2  0  0  0  0
           2  0  0  0  0                     0  2  0  0  2
           0  1  0  0  1                     1  0  1  0  0
           0  1  0  0  1                     1  0  1  0  0
```

Since $b_{31} = 2$, we see that there are 2 ways to go from City #3 (Honolulu) to City #1 (Tokyo) using exactly two flights. From Figure 1, we see that these two routes are: #3→#5→#1 (with a stop in Guam) and #3→#4→#1 (with a stop in Manila). Similarly, $b_{12} = 0$ means that there is no way to go from City #1 (Tokyo) to City #2 (San Francisco) using exactly two flights (check Figure 1 to convince yourself), although there is a direct flight. The entry $c_{12} = 1$ indicates that there is precisely one route that goes from City #1 (Tokyo) to City #2 (San Francisco) and that uses exactly three flights. The reader can verify using Figure 1 that this unique route is:

#1(Tokyo)→#5(Guam)→#1(Tokyo)→#2(San Francisco),

demonstrating that these matrices count literally all possibilities. Of course, programs can be easily designed to weed out such inefficient routes.

QUESTION 2: The owner of the (or any) shipping network would like to know the worst (maximum) number of flights we would actually need to be able to ship from any city to any other city. We define this number (a positive integer) to be the **worst-case-scenario number** of the directed network. How can this be computed?

In case no student raises their hands to offer an answer, ask the following follow-up question that should lead them in the right direction.

QUESTION 3: What do the entries of the matrix $D = A + A^2$ tell us in terms of the shipping network?

Answer: Since the entries of D simply sum the corresponding entries of A (so the number of direct flights) and A^2 (the number of 2-segment flights), the entries of D thus tell us the number of ways to go from row city to column city using either one or two flights, or **at most two** flights.

For example, we compute $D = A + A^2$, and then interpret d_{31}.

```
>> D=A+B
→D=       1  1  1  0  1
          0  0  1  1  1
          2  0  0  1  1
          1  1  0  0  1
          1  1  0  0  1
```

Here, $d_{31} = a_{31} + b_{31} =$ 0 (no direct flights) + 2 (2 two-flight routes indicated above) = 2.

By the same token, it follows that the entries of the matrix $S = A + A^2 + \cdots + A^M$ indicate the number of ways to go from the row city to the column city using **at most *M*** flights. This leads to the following algorithm for answering Question 2, which is easily programmed into an M-file.

Algorithm: Computation of Worst-Case-Scenario Number of a Directed Network:[6]

Step 1: Form the incidence matrix *A*. If all nondiagonal entries are 1, this means there is a direct flight between any two pairs of cities and the worst-case-scenario number of the network is 1. Otherwise, move on to Step 2.

Step 2: (*Iterative Step*) Continue adding increasing powers of *A*: $A + A^2$, $A + A^2 + A^3, \ldots$ until we obtain a matrix that has no nonzero entries except possibly on the main diagonal. The resulting exponent *M* of the last power added will be the worst-case-scenario number.

We point out that since the zeroth power of (any square) matrix *A*, A^0, is defined to be the identity matrix *I*, we can extend the network interpretation of powers of the incidence matrix *A* to include the zeroth power. (With zero flights, we can only go from a city to itself.) With this extended interpretation, the above algorithm can be slightly simplified if we include the zeroth power of *A* in the sums; this allows us to stop as soon as ALL of the matrix entries are zero.

As an example, let's now compute the worst-case-scenario number of the directed network of Figure 1. We have already computed A and $A + A^2$; neither satisfies the nonzero requirements, so we continue with the above algorithm:

```
>> A+A^2+A^3, ans+A^4
→           1  2  1  1  3            →            4  2  2  1  3
ans =       2  0  1  1  1            ans =        2  2  1  1  3
            2  2  0  1  3                         4  2  2  1  3
            2  1  1  0  1                         2  2  1  1  3
            2  1  1  0  1                         2  2  1  1  3
```

Thus we find the worst-case-scenario number of the network is 4. From the second-to-last matrix, we see that there is only one ordered pair of cities (#5 → #4) that would need a full four flights since the row 5 column 4 entry is the only nondiagonal zero entry. The reader can verify with Figure 1 that there is no way to fly from City #5 (Guam) to City #4 (Manila) using less than four flights in this network.

[6] We assume that the network is **connected**, meaning that any city is reachable from any other city. This assumption gives an upper bound of *n* – 1 for the worst-case-scenario number of an *n*-city network (Why?).

QUESTION 4: (a) The owner of the shipping company of Figure 1 would like to investigate the possibility of adding a new flight to the existing schedule (one new arrow) in such a way that the worst-case-scenario number could be brought down as much as possible. Can the worst-case-scenario number be brought down in this way and if so what would be the possible flights to add that would bring it down as much as possible? (b) Answer the same question for two new flights.

Answer: Part (a): There is a simple strategy here. Adding a single new flight to the network simply corresponds to a new incidence matrix with one of the previous zero nondiagonal entries being changed to one. We can simply compute the new worst-case-scenario numbers of all such modified incidence matrices and keep track of those that are less than 4 (the worst-case-scenario number of the original network that was computed in the preceding example). The ones that are as low as possible are the ones that we are interested in. So that we can solve not only this question, but the same one on any network, it is useful to write a MATLAB program to accomplish this task. We apply such an (easily written) program to the incidence matrix under consideration.

(MATLAB input:) `>> AddOneFlt(A)`

(MATLAB output:)

```
By adding one new flight, the worst-case-scenario number of the network    →ans =        1   3
can be reduced from 4 to 3. The segments that will achieve this                           1   4
reduction are stored as the output matrix.                                                2   4
                                                                                          5   2
                                                                                          5   3
                                                                                          5   4
```

Thus, the worst-case-scenario number can be reduced to (as low as) 3 by adding one new link and there are six ways to achieve this reduction. These six links are listed above. For example, the first one: [1 3] corresponds to a new direct flight being added from City #1 (Tokyo) to City #3 (Honolulu).

Part (b): The strategy here is similar. Adding two new flights corresponds to changing two zero nondiagonal entries of the incidence matrix to ones. The worst-case-scenario numbers now need to be checked. We need only record the ones whose worst-case-scenario numbers are less than 3 (the lowest it could get from using one new flight). An M-file can be written in a similar fashion to that used above, but now we will need four nested “for loops” to run through all of the possible ways of adding two new flights (each new flight requires two indices). Once this is done, and the reader runs it on the data of this problem, he/she will find that the worst-case-scenario number cannot be reduced below 3 for any two new flights that are added.

QUESTION 5: Are the above programs practical for general (larger) network routing problems?

Answer: The simplest way to start to understand this question is to run the programs with larger networks. What happens if we try to answer the above questions for the 23-city network shown in Figure 2?

Running the (MATLAB) programs on a PC, we would promptly find the worst-case-scenario number to be eight and that by adding one new flight it could be lowered to as low as seven (in 63 different ways). To find that with two new flights it could be further lowered to six (in 22 different ways), the program starts to take a while to execute, and you would probably want to leave it running while you went out to lunch. If you tried to find out how much lower the worst-case-scenario number could be brought down to using three new flights, you would need to let the computer work over a long weekend. The programs for adding new flights mentioned above, although theoretically sound, are rather brute-force in nature since they simply recomputed the worst-case-scenario number for all of the new possibilities. It thus becomes important to analyze our algorithms and make them as efficient and elegant as possible.

These issues lead us to the very important concept of **complexity** of an algorithm. Complexity measures the approximate size of the amount of computation needed to run an algorithm with input size n, and is usually expressed in the form of a **big-*O* estimate**, $O(f(n))$, where $f(n)$ is a “simple” function of n. To

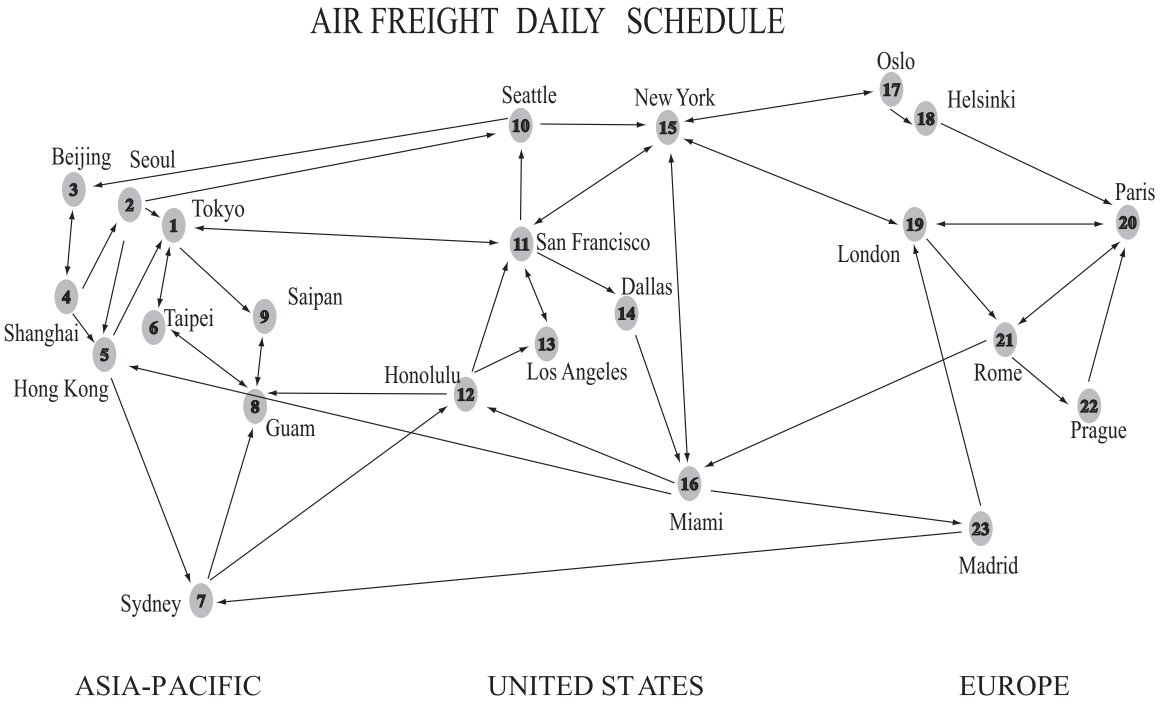

FIGURE 2: A moderately-sized shipping network.

say an algorithm works in $O(f(n))$-time means that with an input of size n, it will take less than $C \cdot f(n)$ logical (or arithmetic) operations, where C is some positive constant. Ideally, we seek big-*O* estimates that are sharp in the sense that examples of the problem (with input size *n*) can be found that will take more than $c \cdot f(n)$ logical operations, for another positive constant *c* (and thus no smaller function can be used in the big-*O* estimate). An algorithm is said to run in **polynomial time** if it admits a big-*O* estimate of the form $O(n^p)$ for some positive integer *p*. Polynomial time algorithms are considered to be good algorithms, but, of course their runtimes can increase significantly for larger exponents *p*, as well as a large value of the (invisible) constant *C*. Each of the algorithms mentioned for the above shipping route problems run in polynomial time, but some of them are still too slow for even moderately sized networks. (The exponents *p* increase quickly with the number of new flights we look into adding to the network.) Many important problems (e.g., the traveling salesman problem) do not yet have algorithms that can even run in polynomial time. The biggest single issue in algorithms is to design them to function as efficiently as possible. For an advanced class that has learned to program in MATLAB, one could end this lesson on network routing with a (difficult) question of the following sort (ideally given as an extra-credit problem):

QUESTION 6: Assume that the following costs (for shipping a 100 lb parcel) are associated with flights in the the 23-city network of Figure 2: Flights within any of the three regions (Asia-Pacific, United States, and Europe) cost $100. Flights from one region to an adjacent region (i.e., Asia-Pacific to/from US, or US to/from Europe) cost $200 and the Madrid to Sidney flight costs $300. The only exceptions are that flights between Guam and Honolulu or Guam and Taipei cost $300. Also, for each such shipment, the company must pay a $2 airport tax to each airport used. Using this information, create a MATLAB M-file: `[price routes] = LowestPrice(i,j)` The two input variables, `i` and `j`, denote the numbers of the origin and destination cities, respectively, for shipping a parcel. The first output variable, `price`, will be the lowest possible price for shipping the parcel, and the second output variable, `routes`, will be a matrix of the corresponding routes to achieve this lowest price (each such route will have the same number of fights–Why?). For example, if you run `LowestPrice(19,22)`, the output should be price = 206, and `routes = [19 21 22]`, indicating that the cheapest way to ship from London to Prague would be to do through Rome and the cost for such a shipment (for one standard parcel) would be \$206 ($=2\times\$100$ (airfares) + $3\times\$2$ (taxes)). Your program should be able to run in less than 5 seconds (on any decent computer) for any pair of inputs.

The simple price structure in this problem allows for a variety of creative thinking and solutions. Initially, this problem might seem much more complex than that of adding one or two good flights. It turns out, however, that a very efficient algorithm exists for finding lowest prices in any such network. This algorithm was discovered by Edsger W. Dijkstra (1930–) [7], a Dutch computer scientist in the 1950s. Dijkstra's algorithm runs in $O(n^2)$-time, where n is the number of cities. It can be geometrically described as follows: Staring at the origin city #*i*, we release a green fluid (representing cost) that flows along unused edges at a rate of \$1/hour. This continues until all edges have been saturated (assuming the network is connected). The minimum cost of getting from city #*i* to ANY other city #*j* is simply the time it takes for the green fluid to reach vertex #*j*. Shortest paths can be kept track of in a straightforward fashion (by tracking predecessor cities). Dijkstra's algorithm remains one of the best for determining minimum costs in general network problems.

REFERENCES:

[Mak-02] Mackiw, George, Experiments with finite linear groups using MATLAB, in *Innovations in Teaching Abstract Algebra* (MAA Notes, #60), Edited by Ellen Maycock and Allen C. Hibbard, Mathematical Association of America, Washington, DC (2002)

[Mat-05] MathWorks' continuously updated website containing descriptions of MATLAB based textbooks (currently contains over 800 titles): `www.mathworks.com/support/books/`

[Pol-03] Polking, John, *Ordinary Differential Equations Using MATLAB, Third Edition*, Prentice-Hall, Upper Saddle River, NJ (2003)

[Ros-02] Ross, Sheldon, *Simulation, Third Edition*, Academic Press, San Diego, CA (2002)

[Sta-04] Stanoyevitch, Alexander, Minimizing lost time at automobile intersections, *Proceedings of the Summer Simulation Conference (2004)*, 48-53, Edited by Agustino G. Bruzzone and Edward Williams, The Society for Modeling and Simulation International, San Diego, CA (2004)

[Sta-05a] Stanoyevitch, Alexander, *Introduction to Numerical Ordinary and Partial Differential Equations Using MATLAB*, John Wiley & Sons, Hoboken, NJ (2005)

[Sta-05b] Stanoyevitch, Alexander, *Introduction to MATLAB with Numerical Preliminaries*, John Wiley & Sons, Hoboken, NJ (2005)

[7] Edsger W. Dijkstra (1930–) is a Dutch computer scientist who studied physics at the University of Amsterdam where he obtained all of his university education and his PhD in 1959. He became very interested in computer programming during his studies well before this was a recognizable line of work. In fact, in 1957, when he put computer programming as his profession in his marriage license, the authorities would not accept it. (He was still able to get married, but changed his profession to physicist.) He became a professor of mathematics at the Eindhoven University of Technology (in the Netherlands) in 1962 until 1973 when he moved to become a research fellow for the Burrows Corp. In 1984, he moved to the US to become a professor of computer science at the University of Texas at Austin, where he remains today.