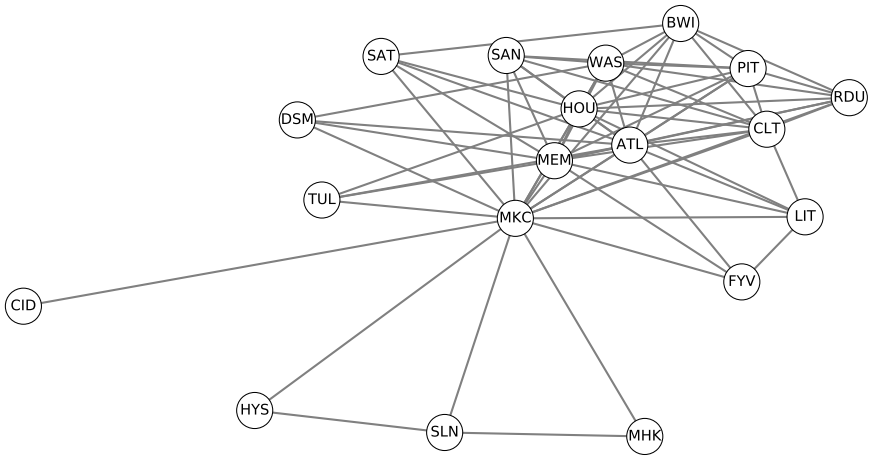
Math 551 Fall 2018
Lab 4

Goals: You will use Matlab to analyze a worldwide air transportation network[1]. The data are over a decade old by now, but still usable for our purposes. In the vocabulary of Section 2.4, the network is represented as an undirected graph with cities (not airports) represented as vertices and with an edge connecting two vertices if it is (or was at the time the data were collected, rather) possible to fly between the two cities. Below is a small portion of this graph, showing some of the cities reachable in two hops from Manhattan at the time. From here, you can get a sense of the age of the data. Note that the vertices are not intended to indicate geographic locations; the only information the graph is intended to convey is the presence or absence of a connecting flight between cities.



In order to treat this graph using linear algebra, each city is assigned a number in the range $1, 2, \ldots, n$ where n is the number of cities in the network. We then define the *adjacency matrix* $\mathbf{A}$ with entries

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

During the lab, you will load the full 3883×3883 adjacency matrix for this network into Matlab and use techniques from linear algebra to analyze it.

Getting started: You will need to download three files and place them into your working directory:

- `m551lab04.m`: the M-file you will edit during the lab
- `global-net.dat`: the network data file to be read by Matlab
- `global-cities.csv`: a spreadsheet translating vertex numbers to city information

What you have to submit: The file `m551lab04.m`, which you will modify during the lab session.

[1] Details of how the network was formed are found here: `http://seeslab.info/downloads/air-transportation-networks/`

Reminders About the Grading Software

Remember, the labs are graded with the help of software tools. If you do not follow the instructions, you will lose points. If the instructions ask you to assign a value to a variable, you must **use the variable name given in the instructions**. If the instructions ask you to make a variable named `x1` and instead you make a variable named `x` or `X` or `X1` or any name other than `x1`, the grading software will not find your variable and you *will* lose points. Required variables are listed in the lab instructions in a gray box like this:

Variables: `x1`, `A`, `q`

At times you will also be asked to answer questions in a comment in your M-file. You must **format your text answers correctly**. Questions that you must answer are shown in gray boxes in the lab. For example, if you see the instruction

Q7: What does the Matlab command `lu` do?

your file must contain a line similar to the following somewhere

```
% Q7: It computes the LU decomposition of a matrix.
```

The important points about the formatting are

- The answer is on a single line.
- The first characters on the line is the comment character `%`.
- The answer is labeled in the form `Q`n`:`, where n stands for the question number from the gray box.

If you do not format your answers correctly, the grading software will not find them and you *will* lose points.

TASKS

Open the file `m551lab04.m` in the Matlab editor and the file `global-cities.csv` in Excel (or other spreadsheet application).

1. In the M-file, run the cell labeled "Load the Network Data." This will load the adjacency matrix `A` and print its dimension (3883×3883). Note that the code uses the function `sparse` to create the matrix. Basically speaking, a sparse matrix is a matrix with many 0s. (We'll see just how sparse `A` is in the next step.) For general matrices, Matlab stores a matrix

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

   as a big block of numbers in memory:

a_{11}	a_{21}	a_{31}	$\cdots$	a_{m1}	a_{12}	a_{22}	a_{32}	$\cdots$	a_{m2}	$\cdots$	a_{mn}

This is called column-major ordering, since the first column is stored in a contiguous block, followed by the second column, and so on. So an $m \times n$ matrix requires mn blocks of memory for internal storage. When many of these entries are 0, it can be much more efficient to store only the nonzero matrix entries. Matlab's internal representation of sparse matrices is beyond the scope of this lab, but the essential idea is that we need to store just 3 pieces of information for each nonzero entry: the i and j indices of the entry and the associated value a_{ij}. If the number of nonzero entries is much smaller than mn, it is often more efficient to use Matlab's sparse matrix functionality. You'll practice some of these functions in the next step. In fact, although Matlab is good at hiding this fact from you, you will benefit from the sparse matrix structure throughout the lab. Many of the standard Matlab functions have special optimized behavior for sparse matrices. So, when you square the matrix `A` later, you'll be taking advantage of this optimized code without even having to think about it.

2. At the bottom of the "Load the Network Data" cell use the functions `numel` and `nnz` to define the variables `numel_A` (the total number of entries in `A`) and `nnz_A` (the number of nonzero entries in `A`). (Remember, if you don't know how to use a function, you can always use `doc`.) Also define the variable `frac_nz_A = nnz_A/numel_A`, which is the fraction of nonzero entries in `A`. If you examine the value of this variable, you should see that only about 0.19% of the entries of `A` are nonzero.

   Variables: `numel_A`, `nnz_A`, `frac_nz_A`

3. In graph theory, the *degree* of a vertex refers to the number of edges connected to it. In the present context, this is the number of different cities that can be reached from a particular city by a direct flight. Since the value a_{ij} is 1 of there is a flight between i and j and 0 otherwise, it is possible to compute the degree of city i as $a_{i1} + a_{i2} + \cdots + a_{in}$. In Matlab, we can do this with the `sum` command. In the section of the M-file titled "Analyzing Degree," you'll see the assignment `deg = sum(A,2)`. This tells Matlab to sum the matrix `A` along the second index (j). In other words, the output is a column vector whose ith entry is the sum of the ith row of `A`.

   By looking in the spreadsheet, we can see that Kansas City is associated with index 1963. If you run this cell in the M-file and examine the value of `deg(1963)` in the Matlab command window, you'll see that Kansas City is connected by direct flight to 59 other cities. The M-file cell also creates a vector of sorted degrees and plots them (with a logarithmic scale on the vertical axis). Look at the plot and notice the rapid growth at the tail. Most cities have few direct connections and few cities have many connections.

   At the bottom of the M-file cell, define the variables `mhk_ind` and `par_ind` to be the indices associated with Manhattan, KS and with Paris, France respectively. (You'll need to look at the spreadsheet.) Then, use these two values to define the variables `deg_mhk` and `deg_par` as the number of direct-flight connections from each of these cities respectively. Manhattan has 2 connected cities, Paris has hundreds.

   Variables: `mhk_ind`, `par_ind`, `deg_mhk`, `deg_par`

4. Now, let's use the interesting fact about the adjacency matrix $\mathbf{A}$ that its powers tell us about *walks* between vertices. More specifically, as described in Section 2.4, the (i, j) entry of $\mathbf{A}^p$ tells us the number of different ways of moving from i to j in p "hops." In the current context, the (i, j) entry of $\mathbf{A}$ gives the number of edges (either 0 or 1) between cities i and j. The (i, j) entry of $\mathbf{A}^2$ counts the number of cities k for which there is a direct connection between i and k and a direct connection between k and j. The (i, j) entry of $\mathbf{A}^3$ counts the number of pairs (k, l) for which there exist direct flights $i \to k \to l \to j$, and so on.

Move to the M-file cell titled "Reachable Cities" and define the variables `A2` and `A3` as the powers $\mathbf{A}^2$ and $\mathbf{A}^3$ respectively. (Note: Since you are already computing $\mathbf{A}^2$ it is more efficient to think of $\mathbf{A}^3 = \mathbf{A}\mathbf{A}^2$ than to simply cube $\mathbf{A}$.)

Variables: `A2`, `A3`

5. In the Matlab command window, enter the following commands

```
>> nnz(A(mhk_ind,:))

ans =

     2

>> nnz(A2(mhk_ind,:))

ans =

    60

>> B = A+A2; nnz(B(mhk_ind,:))

ans =

    60
```

Think about why the following statements are true.

- The first command counts the number of cities reachable from Manhattan by direct flight.
- The second command counts the number of cities reachable from Manhattan using exactly two flights. (Note that Manhattan itself is included in this list.)
- The third command counts the number of cities reachable from Manhattan using *at most* two flights.

At the end of the M-file, define the variables `mhk_3_hop` and `par_3_hop` to be the number of cities reachable using at most three flights from Manhattan and from Paris respectively. You'll need to define a new matrix `B` and then use the `nnz` function. If you get it right, the second variable should be about 4.9151 times as large as the first.

Variables: `mhk_3_hop`, `par_3_hop`