Previous Lecture:
- 2-d array—matrix

Today’s Lecture:
- More examples on matrices
- Characters and strings
- Optional reading: contour plot (7.2, 7.3 in *Insight*)

Announcement:
- Lecture topics re-arranged for the next few weeks
- Project 4 Part A to be posted Friday. Both Parts A and B will be due April 7th.
Matrix example: Random Web

- N web pages can be represented by an N-by-N Link Array $A$.
- $A(i,j)$ is 1 if there is a link on webpage $j$ to webpage $i$.
- Generate a random link array and display the connectivity:
  - There is no link from a page to itself.
  - If $i \neq j$ then $A(i,j) = 1$ with probability $\frac{1}{1+|i-j|}$.
    - There is more likely to be a link if $i$ is close to $j$. 


Random web

N = 20

01110000010010000000
1001000111000000100
0101000000000000000
0010100000000000000
0001000000110000000
000000000000010100000
01111100010110000000
00000010000100000011
0100000001001000100
00000001101000000001
00000010000011000000
00000010010000000001
001000011010101100000
000000100000001100000
000001010000010010001
00000010001000010100
0100000010001010110
0000000000000011001
00000010000000000000
00000000000000010100
function A = RandomLinks(n)
% A is n-by-n matrix of 1s and 0s
% representing n webpages

A = zeros(n,n);
for i=1:n
    for j=1:n
        r = rand;
        if i~=j && r <= 1/(1 + abs(i-j));
            A(i,j) = 1;
        end
    end
end
Represent the web pages graphically…

100 Web pages arranged in a circle. Next display the links….
Bidirectional links are blue. Unidirectional link is black as it leaves page j, red when it arrives at page i.
for i = 1:n
    for j = 1:n
        end
    end
end
Somewhat inefficient: each blue line gets drawn twice.

See ShowRandomLinks.m
Transpose—like switching row and column indices
% Given an n-by-m matrix A.
% What is this operation?
for g = 1: n
    for h = 1: floor(m/2)
        A(g,h) = A(g, m-h+1);
    end
end

Reflect the right half of A onto the left half

Reflect the bottom half of A onto the top half
% Given an nr-by-nc matrix A.
% What is this operation?
for  r= 1: nr
    for  c= 1: floor(nc/2)
        A(r,c)= A(r, nc-c+1);
    end
end

a. Reflect the right half of A onto the left half
b. Reflect the bottom half of A onto the top half
Accessing a submatrix

- $M$ refers to the whole matrix
- $M(3,5)$ refers to one component of $M$

![Matrix](image)
Accessing a submatrix

- \( \textbf{M} \) refers to the whole matrix
- \( \textbf{M}(3,5) \) refers to one component of \( \textbf{M} \)
- \( \textbf{M}(2:3,3:5) \) refers to a submatrix of \( \textbf{M} \)

![Matrix with submatrix highlighted]
Accessing a submatrix

- \( \mathbf{M} \) refers to the whole matrix
- \( \mathbf{M}(3,5) \) refers to one component of \( \mathbf{M} \)
- \( \mathbf{M}(2:3,3:5) \) refers to a submatrix of \( \mathbf{M} \)
- \( \mathbf{M}(2,:) \) refers to row 2 of \( \mathbf{M} \)
Accessing a submatrix

- \( M \) refers to the whole matrix
- \( M(3, 5) \) refers to one component of \( M \)
- \( M(2:3, 3:5) \) refers to a submatrix of \( M \)
- \( M(2, :) \) refers to row 2 of \( M \)
- \( M(:, 3:5) \) refers to columns 3-5 of \( M \)

All indices
Characters & strings

- We have used strings already:
  - `n= input('Next number: ')`
  - `sprintf('Answer is %d', ans)`
- A string is made up of individual characters, so a string is a 1-d array of characters
- ‘CS1112 rocks!’ is a character array of length 13; it has 7 letters, 4 digits, 1 space, and 1 symbol.

`['C' 'S' '1' '1' '1' '2' 'r' 'o' 'c' 'k' 's' '!']`

- Can have 2-d array of characters as well

`[['C' 'S' '1' '1' '1' '2' 'r' 'o' 'c' 'k' 's' '!'], ['r' 'o' 'c' 'k' 's' '!']]` 2×6 matrix
Matlab types: \texttt{char}, \texttt{double}, \texttt{uint8}, \texttt{logical}

\begin{itemize}
  \item \texttt{a} is a 1-d array with type \texttt{char} components. We call \texttt{a} a “string” or “char array”
  \item \texttt{b} is a 1-d array with type \texttt{double} components. \texttt{double} is the default type for numbers in Matlab. We call \texttt{b} a “numeric array”
  \item \texttt{c} is a 1-d array with type \texttt{uint8} components. We call \texttt{c} a “uint8 array”
  \item \texttt{d} is a scalar of the type \texttt{logical}. We call \texttt{d} a “boolean value”
\end{itemize}
### Some Matlab types: `char`, `double`, `logical`

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>char</code></td>
<td><code>'C' 'S' '1'</code></td>
<td><code>a</code> is a 1-d array with type <code>char</code> components. We call <code>a</code> a “string” or “char array”</td>
</tr>
<tr>
<td><code>double</code></td>
<td><code>[3 9]</code></td>
<td><code>b</code> is a 1-d array with type <code>double</code> components. <code>double</code> is the default type for numbers in Matlab. We call <code>b</code> a “numeric array”</td>
</tr>
<tr>
<td><code>logical</code></td>
<td><code>rand &gt; .5</code></td>
<td><code>d</code> is a scalar of the type <code>logical</code>. We call <code>d</code> a “boolean value”</td>
</tr>
</tbody>
</table>

There is not a type “string”! What we call a “string” is a 1-d array of chars.
Single quotes enclose strings in Matlab

Anything enclosed in single quotes is a string (even if it looks like something else)

- ‘100’ is a character array (string) of length 3
- 100 is a numeric value
- ‘pi’ is a character array of length 2
- pi is the built-in constant 3.1416…
- ‘x’ is a character (vector of length 1)
- x may be a variable name in your program
Strings are vectors

Vectors

- **Assignment**
  
  \( v = [7 \ 0 \ 5]; \)

- **Indexing**
  
  \( x = v(3); \quad \% x \text{ is } 5 \)
  
  \( v(1) = 1; \quad \% v \text{ is } [1 \ 0 \ 5] \)
  
  \( w = v(2:3); \quad \% w \text{ is } [0 \ 5] \)

- **: notation**
  
  \( v = 2:5; \quad \% v \text{ is } [2 \ 3 \ 4 \ 5] \)

- **Appending**
  
  \( v = [7 \ 0 \ 5]; \)
  
  \( v(4) = 2; \quad \% v \text{ is } [7 \ 0 \ 5 \ 2] \)

- **Concatenation**
  
  \( v = [v \ [4 \ 6]]; \quad \% v \text{ is } [7 \ 0 \ 5 \ 2 \ 4 \ 6] \)

Strings

- **Assignment**
  
  \( s = \text{‘hello’}; \)

- **Indexing**
  
  \( c = s(2); \quad \% c \text{ is } \text{‘e’} \)
  
  \( s(1) = \text{‘J’}; \quad \% s \text{ is } \text{‘Jello’} \)
  
  \( t = s(2:4); \quad \% t \text{ is } \text{‘ell’} \)

- **: notation**
  
  \( s = \text{‘a’}:\text{‘g’}; \quad \% s \text{ is } \text{‘abcdefg’} \)

- **Appending**
  
  \( s = \text{‘duck’}; \)
  
  \( s(5) = \text{‘s’}; \quad \% s \text{ is } \text{‘ducks’} \)

- **Concatenation**
  
  \( s = [s \ ‘ \text{quack’}]; \quad \% s \text{ is } \text{‘ducks quack’} \)
Example: removing all occurrences of a character

- From a genome bank we get a sequence
  ATTG CCG TA GCTA CGTACGC AACTGG AAATGGC CGTAT...

- First step is to “clean it up” by removing all the blanks. Write this function:

```matlab
function s = removeChar(c, s)
% Return string s with all occurrences
% of character c removed
```
Example: removing all occurrences of a character

Can solve this problem using iteration—check one character (one component of the vector) at a time

```matlab
function s = removeChar_loop(c, s)
    % Return string s with all occurrences of character c removed.
```
Example: removing all occurrences of a character

Can solve this problem using iteration—check one character (one component of the vector) at a time

```matlab
function s = removeChar_loop(c, s)
    % Return string s with all occurrences of
    % character c removed.

    t = '';  
    for k = 1:length(s)
        end
    s = t;
```
Example: removing all occurrences of a character

Can solve this problem using iteration—check one character (one component of the vector) at a time

```matlab
function s = removeChar_loop(c, s)
% Return string s with all occurrences of character c removed.

t = '';
for k = 1:length(s)
    if s(k) ~= c
        t = [t s(k)];
    end
end
s = t;
```