- Previous Lecture:
  - Discrete vs. continuous; finite vs. infinite
  - Vectorized operations
- Today's Lecture:
  - Vectorized operations and plots
  - 2-d array—matrix
- Announcements:
  - Prelim 1 tonight at 7:30pm
    - Last names A-H in Olin Hall 255
    - Last names I-L in Olin Hall 245
    - Last names M-Z in Olin Hall 155
  - Fall Break Mon & Tues: no lec, dis, office/consulting hrs. Attendance at 10/10 (W) dis is optional, but the exercise is required. Attend any of the 10/10 dis sections for help if you wish.

Initialize vectors/matrices if dimensions are known

…instead of "building" the array one component at a time

```
% Initialize y
x=linspace(a,b,n);
y=zeros(1,n);
for k=1:n
   y(k)=myF(x);
end
```

```
% Build y on the fly
x=linspace(a,b,n);

for k=1:n
   y(k)=myF(x);
end
```

Much faster for large n!

Vectorized code allows an operation on multiple values at the same time

```
yellow= [1 1 0];
black = [0 0 0];

% Average color via vectorized op
colr= 0.5*yellow + 0.5*black;
```

Operation performed on vectors

Vectorized addition

$$[.5\ .5\ 0] + [0\ 0\ 0] = [.5\ .5\ 0]$$

```
% Average color via scalar op
for k = 1:length(black)
   colr(k)= 0.5*yellow(k) + 0.5*black(k);
end
```

Operation performed on scalars

Vectorized code
—a Matlab-specific feature

See Sec 4.1 for list of vectorized arithmetic operations

- Code that performs element-by-element arithmetic/relational/logical operations on array operands in one step

- Scalar operation: x + y
  where x, y are scalar variables

- Vectorized code: x + y
  where x and/or y are vectors. If x and y are both vectors, they must be of the same shape and length

Vectorized addition

	x	2	1	.5	8
+	y	1	2	0	1
=	z	3	3	.5	9

Matlab code: `z= x + y`

Vectorized subtraction

	x	2	1	.5	8
-	y	1	2	0	1
=	z	1	-1	.5	7

Matlab code: **`z= x - y`**

Vectorized multiplication

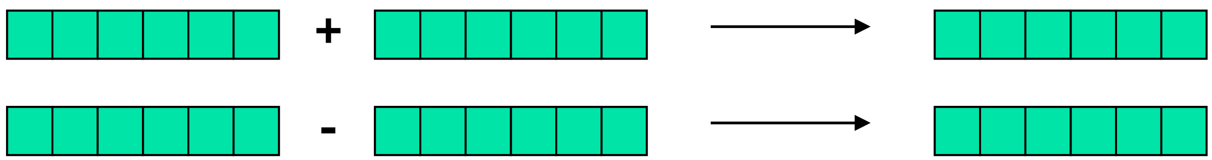
	a	2	1	.5	8
×	b	1	2	0	1
=	c	2	2	0	8

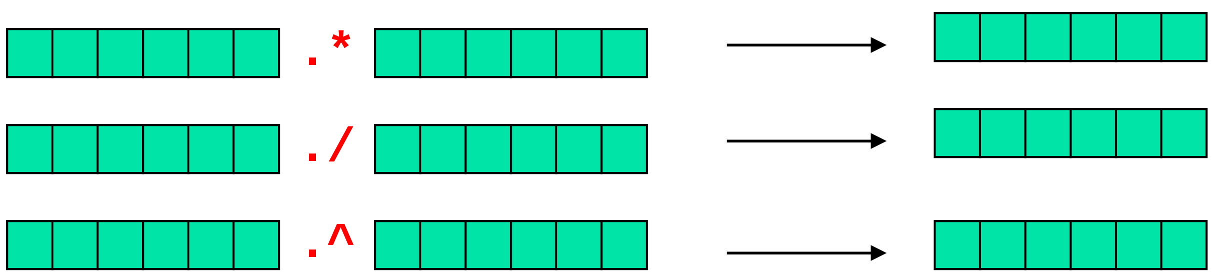
Matlab code: **c= a .* b**

See full list of ops in §4.1

Vectorized

element-by-element arithmetic operations on arrays





A dot (.) is necessary in front of these math operators

Shift

	x	3				
+	y	2	1	.5	8	
=	z	5	4	3.5	11	

Matlab code: `z= x + y`

Reciprocate

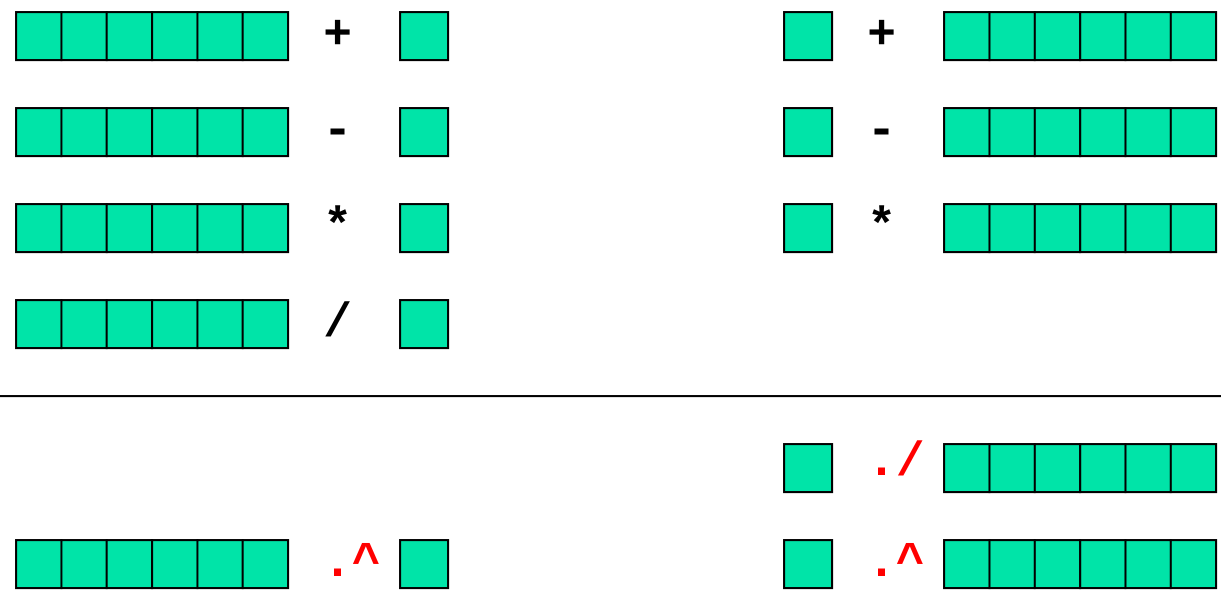
	x	1			
/	y	2	1	.5	8
=	z	.5	1	2	.125

Matlab code: `z= x ./ y`

See full list of ops in §4.1

Vectorized

element-by-element arithmetic operations between an array and a scalar



A dot (.) is necessary in front of these math operators

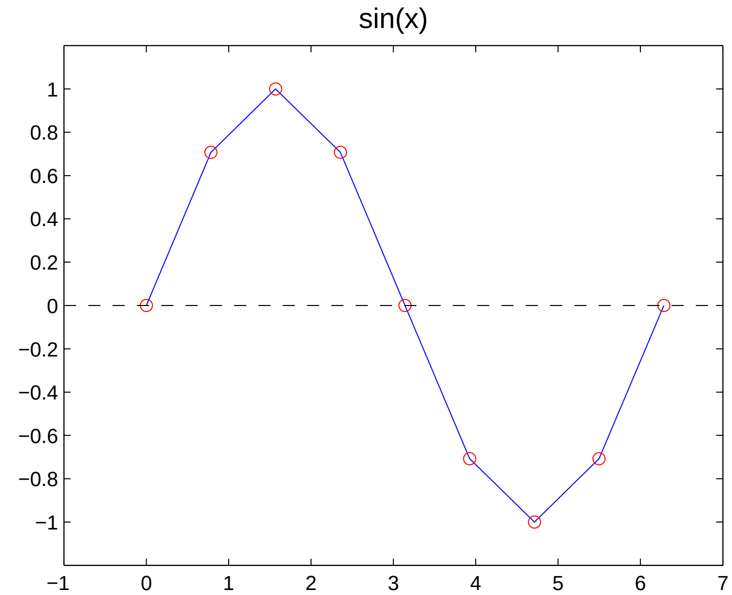
The dot in .* , .* , ./ not necessary but OK

Generating tables and plots

x	sin(x)
0.000	0.000
0.784	0.707
1.571	1.000
2.357	0.707
3.142	0.000
3.927	-0.707
4.712	-1.000
5.498	-0.707
6.283	0.000

x, **y** are vectors. A vector is a 1-dimensional list of values

```
x= linspace(0,2*pi,9);
y= sin(x);
plot(x,y)
```



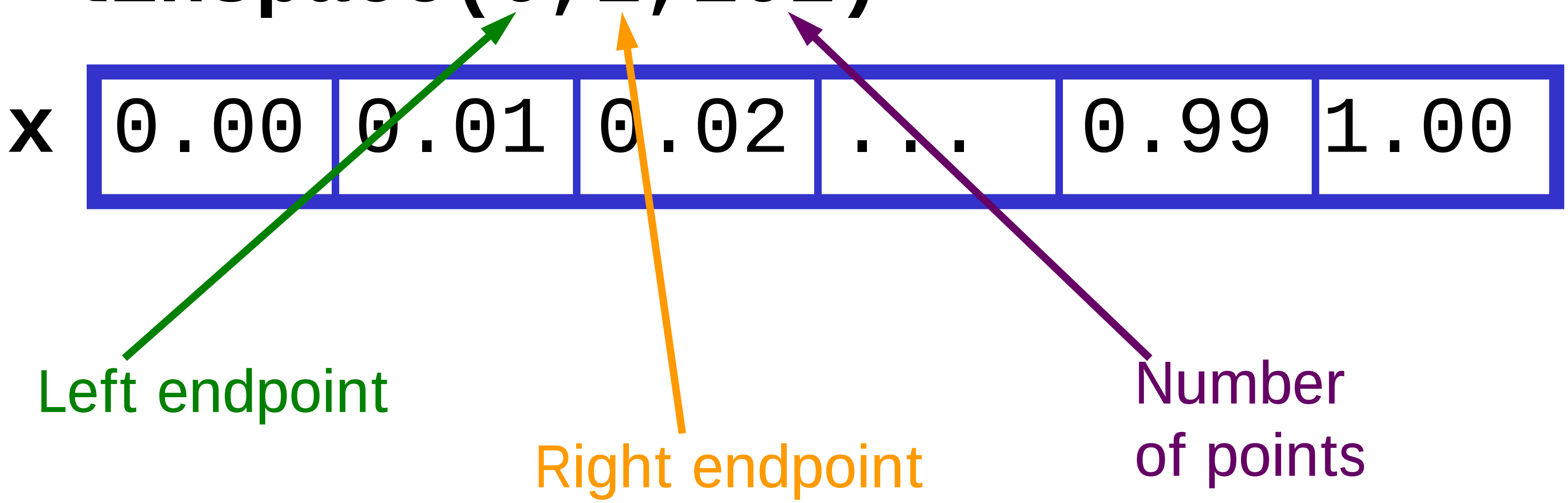
Note: x, y are shown in columns due to space limitation; they should be rows.

Built-in function `linspace`

```
x= linspace(1,3,5)
```

x	1.0	1.5	2.0	2.5	3.0

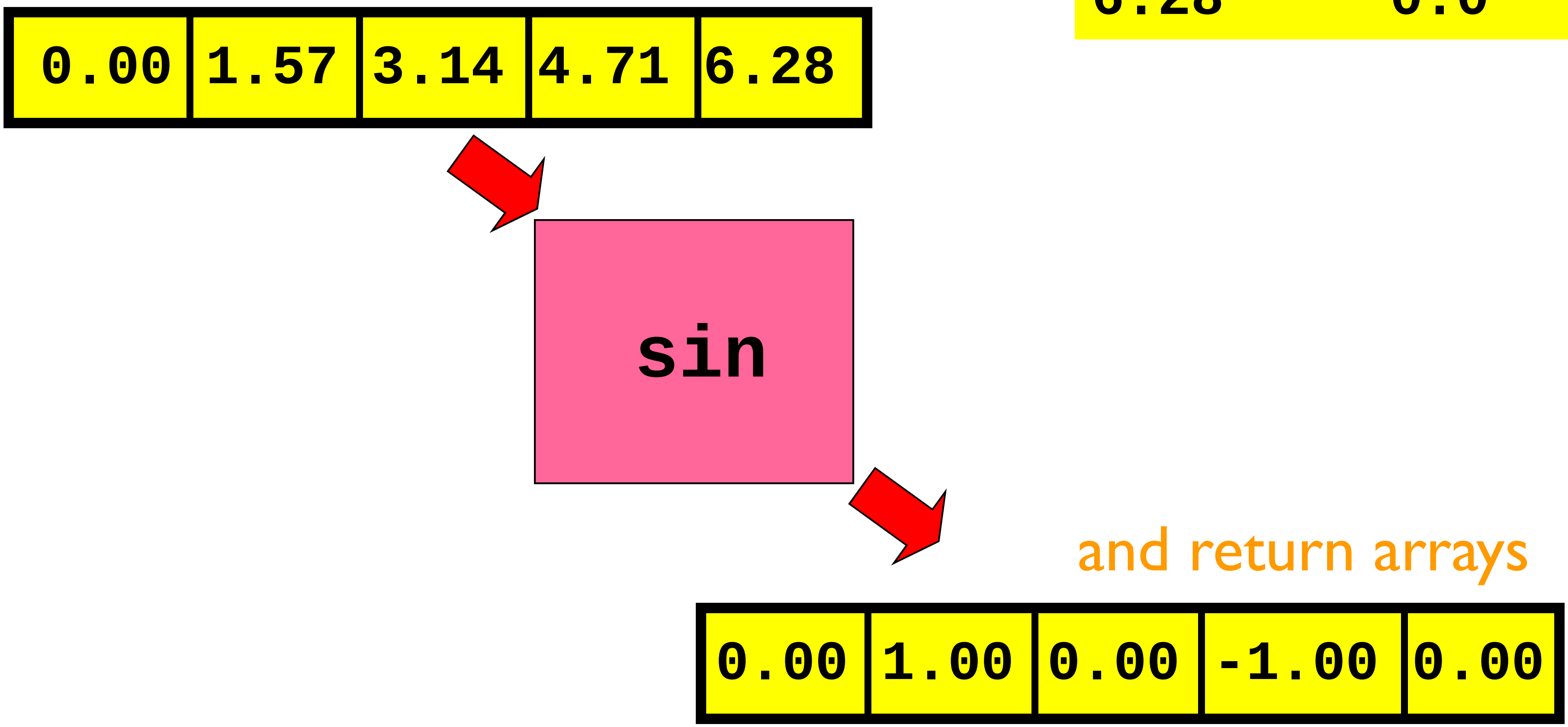
```
x= linspace(0,1,101)
```



How did we get all the sine values?

x	sin(x)
0.00	0.0
1.57	1.0
3.14	0.0
4.71	-1.0
6.28	0.0

Built-in functions accept arrays



Can we plot this?

$$f(x) = \frac{\sin(5x)\exp(-x/2)}{1+x^2}$$

for

-2 <= x <= 3

Can we plot this?

$$f(x) = \frac{\sin(5x)\exp(-x/2)}{1+x^2}$$

for

-2 <= x <= 3

Yes!

See **plotComparison.m**

Can we plot this?

$$f(x) = \frac{\sin(5x)\exp(-x/2)}{1+x^2}$$

for

-2 <= x <= 3

Yes!

```
x = linspace(-2,3,200);
y = sin(5*x).*exp(-x/2)./(1 + x.^2);
plot(x,y)
```

Element-by-element arithmetic operations on arrays

Element-by-element arithmetic operations on arrays… Also called "vectorized code"

```
x = linspace(-2,3,200);
y = sin(5*x).*exp(-x/2)./(1 + x.^2);
```

x and **y** are vectors

Contrast with scalar operations that we've used previously…

```
a = 2.1;
b = sin(5*a);
```

a and **b** are scalars

The operators are (mostly) the same; the operands may be scalars or vectors.

When an operand is a vector, you have "vectorized code."

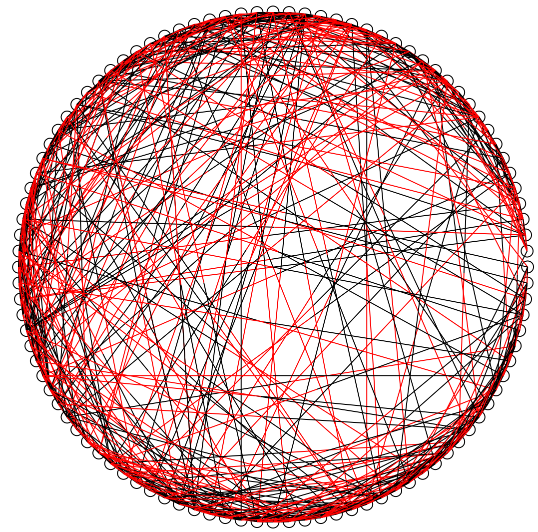
Storing and using data in *tables*

A company has 3 factories that make 5 products with these costs:

C

10	36	22	15	62
12	35	20	12	66
13	37	21	16	59

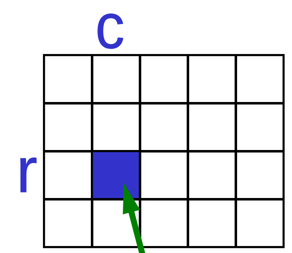
What is the best way to fill a given purchase order?



Connections between webpages

0	0	1	0	1	0	0
1	0	0	1	1	1	0
0	1	0	1	1	1	1
1	0	1	1	0	1	0
0	0	1	1	0	1	1
0	0	1	0	1	0	1
0	1	1	0	1	1	0

2-d array: **matrix**



- An array is a named collection of like data organized into rows and columns
- A 2-d array is a table, called a *matrix*
- Two *indices* identify the position of a value in a matrix, e.g.,

  `mat(r,c)`

  refers to component in row r, column c of matrix mat
- Array index starts at 1
- Rectangular: all rows have the same #of columns

Creating a matrix

- Built-in functions: ones, zeros, rand
  - E.g., zeros(2,3) gives a 2-by-3 matrix of 0s
- "Build" a matrix using square brackets, [ ], but the dimension must match up:
  - [x y] puts y to the right of x
  - [x; y] puts y below x
  - [4 0 3; 5 1 9] creates the matrix

4	0	3
5	1	9

  - [4 0 3; ones(1,3)] gives

4	0	3
1	1	1

  - [4 0 3; ones(3,1)] doesn't work

Working with a matrix: `size` and individual components

Given a matrix M

2	-1	.5	0	-3
3	8	6	7	7
5	-3	8.5	9	10
52	81	.5	7	2

```
[nr, nc]= size(M)  % nr is #of rows,
                   % nc is #of columns
nr= size(M, 1)  % # of rows
nc= size(M, 2)  % # of columns

M(2,4)= 1;
disp(M(3,1))
M(1,nc)= 4;
```

```
% What will M be?
M = [ones(1,3); 1:4]
```

A

```
1  1  1  0
1  2  3  4
```

B

```
1  1  1
1  2  3
```

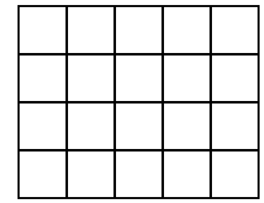
C

Error – M not created

What will A be?

```
A= [0  0]
A= [A'  ones(2,1)]
A= [0  0  0  0;  A  A]
```

Example: minimum value in a matrix



```
function val = minInMatrix(M)
% val is the smallest value in matrix M
```

minInMatrix.m

Pattern for traversing a matrix M

```
[nr, nc] = size(M)
for  r= 1:nr
     % At row r
     for  c= 1:nc
          % At column c (in row r)
          %
          % Do something with M(r,c) …
     end
end
```

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

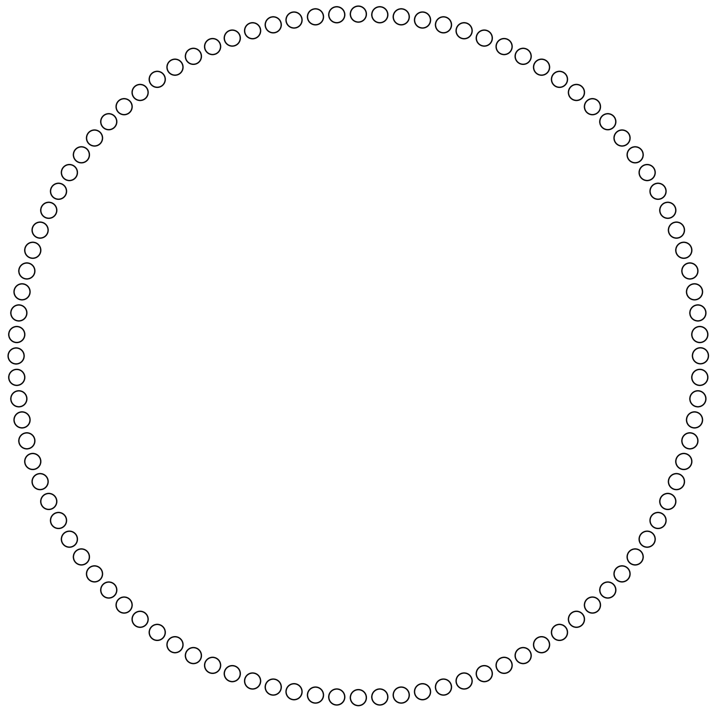
```
function A = RandomLinks(n)
% A is n-by-n matrix of 1s and 0s
% representing n webpages

A = zeros(n,n);
for i=1:n
  for j=1:n
    r = rand(1);
    if i~=j && r<= 1/(1 + abs(i-j));
        A(i,j) = 1;
    end
  end
end
```

Random web

N = 20

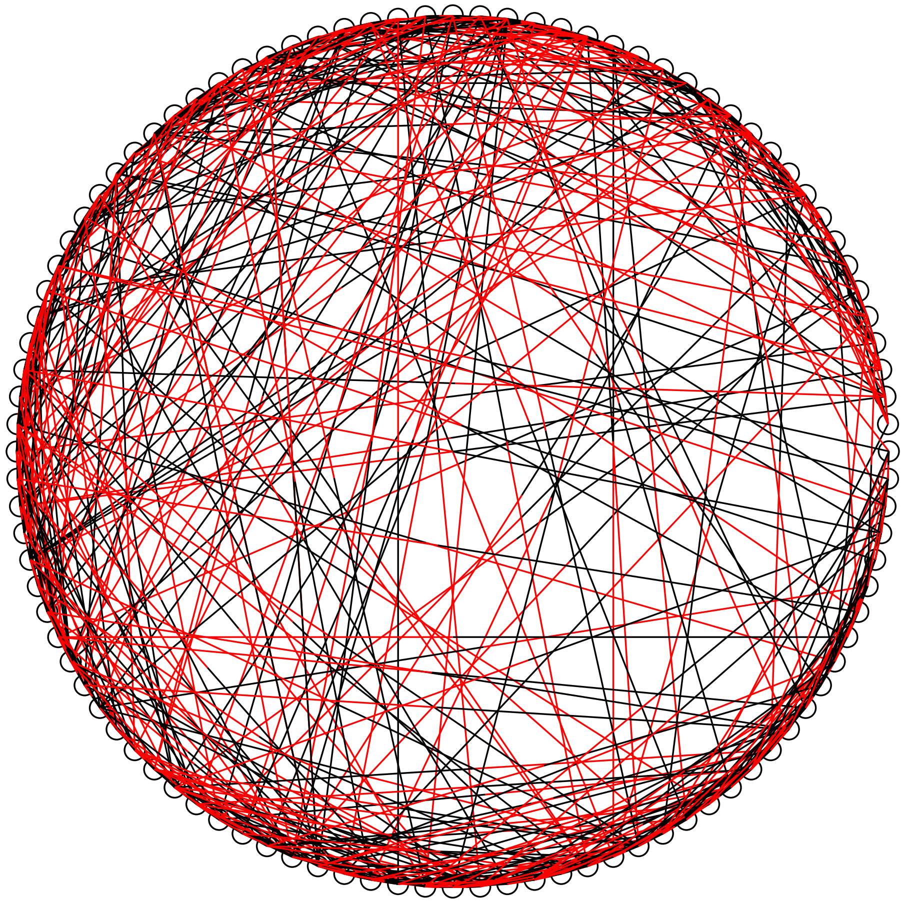
0	1	1	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0	1	0	0
0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0
0	1	1	1	1	1	0	0	0	1	0	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1	1
0	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	1	1	0	1	0	1	1	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	1	0	1	0	0	0	0	1	0	0	1	0	0	0	1
0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	1	0
0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	1	0	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

Represent the web pages graphically…



100 Web pages arranged in a circle.
Next display the links….

Represent the web pages graphically…



Line black as it leaves page j, red when it arrives at page i

ShowRandomLinks.m