Nodes, adjacency matrix, out degrees

N **nodes** in the network.

Adjacency matrix $A = (a_{ij})$. It has size $N \times N$ and contains entries of 0 or 1.

$$a_{ij} = 1 \quad \text{indicates a link from node } j \text{ to node } i.$$

The sum of the entries in column j is

$$d_j = \sum_{i=1}^{n} a_{ij} = \textbf{out degree of node } j.$$

To cope with out degrees which are 0 let

$$\tilde{d}_j = \begin{cases} d_j, & \text{if } d_j > 0, \\ 1, & \text{if } d_j = 0. \end{cases}$$

Let

$$D = \operatorname{diag}\{\frac{1}{\tilde{d}_1}, \dots, \frac{1}{\tilde{d}_N}\}.$$

The original matrix C – re-written as C_0

The matrix C in earlier sessions without any adjustments is now written as C_0 and can be written in the form

$$C_0 = AD, \quad (C_0)_{ij} = \begin{cases} \dfrac{a_{ij}}{d_j}, & \text{when } d_j > 0, \\ 0, & \text{otherwise.} \end{cases}$$

The simulations in the previous sessions involved starting with a vector $\underline{p}^{(0)}$ with 1 in one position and 0 in the other entries and then generating vectors by

$$\underline{p}^{(k+1)} = C_0 \underline{p}^{(k)}, \quad k = 0, 1, 2, \dots.$$

The entry $\left(\underline{p}^{(k)}\right)_i \ge 0$ indicates at the k stage the probability of being at node i.

Problems with the previous model

- If the out degree $d_j = 0$ then you if you reach node j then you stay at node j.
- If the network contains a closed loop then you stay in the loop.

The modified model (implemented in `cmat2.m`)

$$\underline{p}^{(k+1)} = C \underline{p}^{(k)},$$

with

$$C = (1-\alpha) C_0 + \frac{1}{N-1} \begin{pmatrix} 0 & 1 & 1 & \cdots & 1 \\ 1 & 0 & 1 & \cdots & 1 \\ 1 & 1 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 & 0 \end{pmatrix} \hat{D},$$

where $\alpha \in [0, 1]$ and

$$\hat{D} = \operatorname{diag}(\hat{d}_i), \quad \hat{d}_i = \begin{cases} \alpha, & \text{if } d_i > 0, \\ 1, & \text{if } d_i = 0. \end{cases}$$

The matrix $C = (c_{ij})$ in component form

$$c_{ij} = \begin{cases} 0, & \text{if } i = j, \text{ i.e. on the diagonal,} \\ \dfrac{\alpha}{N-1} + (1-\alpha)\dfrac{a_{ij}}{d_j}, & \text{if } d_j \ne 0, \\ \dfrac{1}{N-1} & \text{if } d_j = 0. \end{cases}$$

Note that the entries in each column sum to 1.

The case $d_j = 0$ is the same as the case $d_j = N - 1$ with each non-diagonal entry being $1/(N-1)$.

The long term probabilities

It can be proved that the sequence of vectors $\underline{p}^{(0)}$, $\underline{p}^{(1)}$, $\underline{p}^{(2)}, \ldots$ converges and if $\underline{p}$ denotes the limit then the **probability vector** $\underline{p}$ satisfies

$$C\underline{p} = \underline{p}.$$

Thus $\underline{p} = (p_i)$ is an **eigenvector** of C with **eigenvalue** 1 and as it is also a probability vector the components satisfy

$$0 \le p_i \le 1, \quad p_1 + p_2 + \cdots + p_N = 1.$$

A method to determine $\underline{p}$

`direct1.m` and `mkw_direct1.m` use the following.

$$\begin{aligned} (C - I)\underline{p} &= \underline{0}, \\ \underline{e}^T \underline{p} &= 1, \end{aligned}$$

where $\underline{e} = (1, 1, \ldots, 1)^T$ is $N \times 1$ vector with each entry equal to 1.

In Matlab this can be done with the following 3 statements.

```
M = [ C - eye(N, N);
      ones(1, N) ];
b = [ zeros(N, 1);
          1 ];
p = M \ b
```

The Matlab operator \ in this case involves the least squares solution of the **over-determined system** of $N + 1$ equations in N unknowns and in this case it also exactly solves the equations.

The function file `mkw_cmat2.m`

```
function C = mkw_cmat2(A, alpha)

[N, M] = size(A);

% get the out-degrees d, i.e. the sum of each column
d = sum(A);

% allocate space for C
C = zeros(N, N);

% set the entries column-by-column
for j=1:N
  % deal with d(j)==0 case and d(j)>0 case
  if d(j)==0
    C(:, j) = 1/(N-1);
  else
    C(:, j) = alpha/(N-1)+(1-alpha)*A(:, j)/d(j);
  end
  C(j, j) = 0;
end
```


The script file `mkw_direct1.m`

```
fprintf('Adjacency matrix for network 7 follows\n')
A = [ 0 1 0 1 ;
      0 0 0 0 ;
      1 1 0 0 ;
      1 1 1 0 ]

alpha = 0.3;
fprintf('probability matrix C when alpha=%f follows\n', alpha);
C = mkw_cmat2(A, alpha)

N = size(A, 2);

M = [ C - eye(N, N);
      ones(1, N) ];
b = [ zeros(N, 1);
          1 ];

% use \ to get p and show p
fprintf('probability vector p follows\n')
p = M \ b
```

Comments about better methods to determine $\underline{p}$

If you search the internet for information about the PageRank algorithm then better techniques are given which are not too much more complicated. When N is large you can only use these better techniques and the key is not to create the full matrix C given by

$$C = (1-\alpha)AD + \frac{1}{N-1}\begin{pmatrix} 0 & 1 & 1 & \cdots & 1 \\ 1 & 0 & 1 & \cdots & 1 \\ 1 & 1 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 & 0 \end{pmatrix}\hat{D}.$$

This can be written as

$$C = (1-\alpha)AD + \left(\frac{1}{N-1}\right)(\underline{e}\,\underline{e}^T - I)\hat{D}.$$

You only need to store A, the diagonal entries of D and $\hat{D}$ and the parameter α in the better approaches.

The product $C\underline{p}$ and $C\underline{p} = \underline{p}$

Now

$$\begin{aligned} C\underline{p} &= (1-\alpha)AD\underline{p} + \frac{1}{N-1}(\underline{e}\,\underline{e}^T - I)\hat{D}\underline{p} \\ &= (1-\alpha)AD\underline{p} + \left(\frac{1}{N-1}\right)(\gamma\underline{e} - \hat{D}\underline{p}), \quad \gamma = \underline{e}^T\hat{D}\underline{p}. \end{aligned}$$

The equation $C\underline{p} = \underline{p}$ can be re-arranged as

$$\left(I + \left(\frac{1}{N-1}\right)\hat{D} - (1-\alpha)AD\right)\underline{p} = \left(\frac{\gamma}{N-1}\right)\underline{e}.$$

This is N equations in N unknowns. Let $\hat{M}$ denote the matrix on the left hand side. As γ involves $\underline{p}$ we solve

$$\hat{M}\underline{y} = \underline{e}$$

and then extract $\underline{p}$ by using

$$r = y_1 + \cdots + y_N, \quad \text{and} \quad p_i = \frac{y_i}{r}, \quad i = 1, \ldots, n.$$

A Matlab implementation

If the adjacency matrix, which may be large and sparse, is already given then a complete solution is as follows.

```
d=sum(A);   d=d(:);
i0=find(d==0);
d(i0)=1;
D=spdiags(1./d, 0, N, N);

% set up the terms in the matrix
C0=(1-alpha)*A*D;
e=ones(N, 1);
dhat=alpha*e;
dhat(i0)=1;
Dhat=spdiags(dhat, 0, N, N);
I=speye(N, N);

% solve with the sparse matrix for p
y=( I+(1/(N-1)*Dhat-C0 ))\e;
p=y/sum(y);
```

You will meet `spdiags()`, `speye()` at level 2 and also possibly the `find()` function.

Cell arrays in Matlab

In the assignment you need a list of topics. An example of a list of names in Matlab is illustrated by the following.

```
topics={'Calculus', 'Algebra',...
        'Matrices', 'Differentiation'};
N=length(topics);
for i=N:-1:1
  disp(topics{i})
end
```

`topics` is a cell array. `topics{i}` refers to the entry in position `i`.

Setting up mixed data

Consider the following set-up in Matlab. The countries are in alphabetical order. The statement which assigns p just contains the numbers.

```
pop={...
'Bangladesh', 162221000,...
'Brazil',     192540000,...
'China',     1336070000,...
'India',     1177592000,...
'Indonesia',  231369500,...
'Japan',      127430000,...
'Mexico',     107550697,...
'Nigeria',    154729000,...
'Pakistan',   168840500,...
'Russia',     141927297,...
'USA',        308765000};

p=cell2mat( pop(2:2:end) );
N=length(p);
```


Sorting the data

With the previous set-up you can sort the data and put in size order instead of alphabetical order as follows.

```
[a, o]=sort(p, 'descend');

for i=1:N
  j=o(i);
  fprintf('%12d %s\n', p(j), pop{2*j-1});
end
```

Part of the output is as follows.

```
  1336070000 China
  1177592000 India
   308765000 USA
  ..........
   127430000 Japan
   107550697 Mexico
```


Summary of Matlab part of the assignment

- For a given network create the adjacency matrix A. Take care that this is correct. You may wish to use the function file `from_to.m`.
- Set α and use one of the methods to get $\underline{p}$.
- Sort the entries in $\underline{p}$ and display the results.

All the deliverables

1. A one A3-page poster. (This can be printed on 2 A4 pages.) Submit to JNCK 101 by 15:30 on Mon 14th March.
2. A version of the script `direct1.m` must be sent by email to me. I will let you know whether or not it runs.
3. Any m-files associated with plotting $\underline{p} = \underline{p}(\alpha)$, $0 \le \alpha \le 1$ should also be sent by email.

The interview

This is scheduled for 14:00–15:00 on Tue 15th March 2016. There will be a group component and an individual component.

The mark obtained will depend on the poster, the m-files submitted and the individual part of the interview.